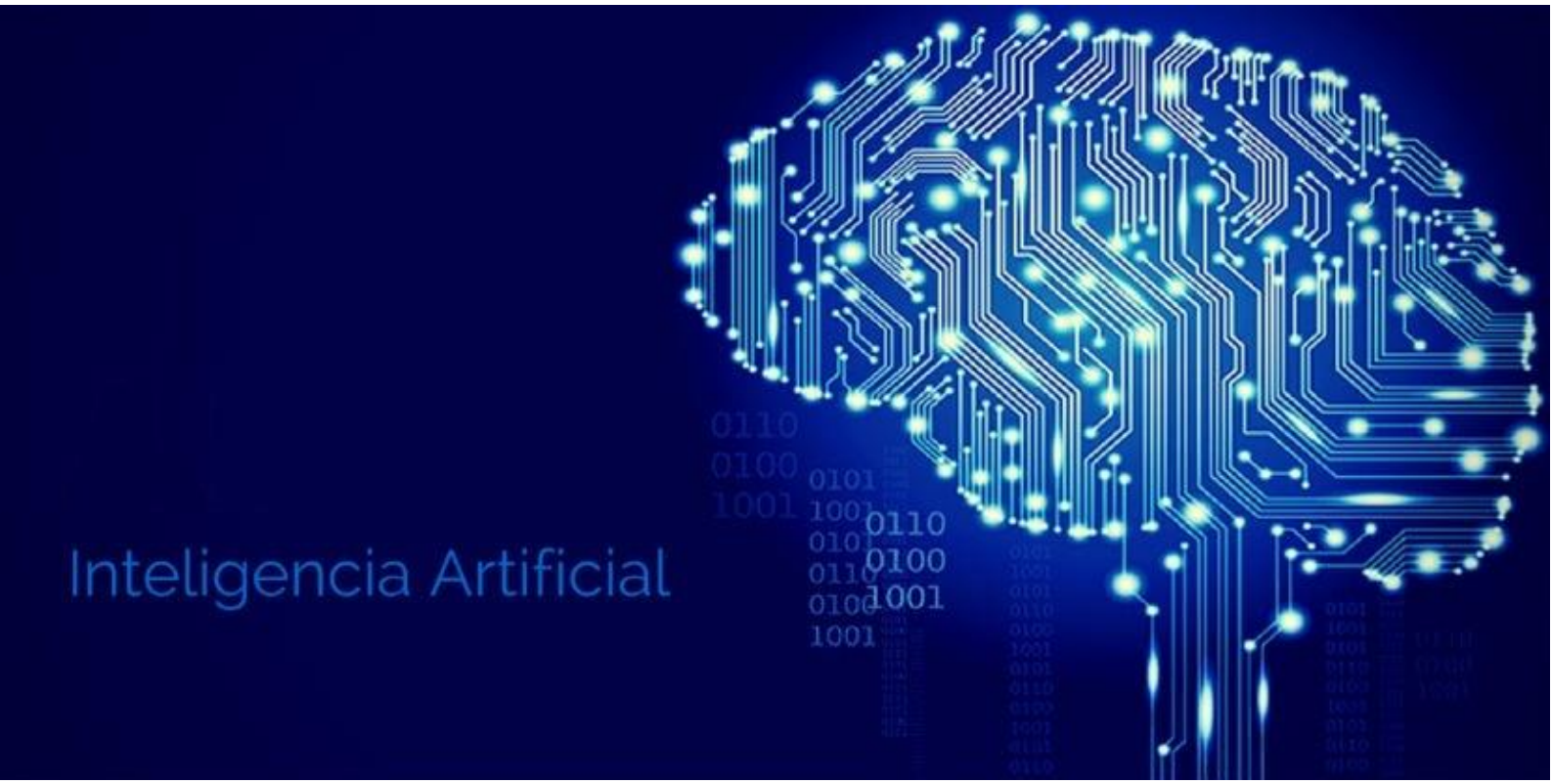
Universidad
Carlos III de Madrid
www.uc3m.es



TRABAJO FIN DE GRADO

Desarrollo de capa de inteligencia artificial en videojuegos

Grado de Ingeniera Informática 2014/2015

Adrián Carruana Martín

Tutor: Yago Sáez

Agradecimientos

Me gustaría agradecer a mi familia por apoyarme y estar siempre detrás de mí para que en ningún momento me relajara. Sobre todo a mis padres por sus recurrentes frases que solo buscaban que les llevara la contraria.

A mi novia, por estar conmigo en los peores momentos, darme los ánimos necesarios para seguir adelante y no rendirme, y sobre todo por comprenderme.

A mis amigos, que siempre pudieron sacarme una sonrisa y ayudarme a despejar la mente para volverla a llenar.

A mis compañeros de universidad, por esa gran lucha que hemos realizado juntos llamadas prácticas y por la mejor compañía que he tenido en una clase.

A mi tutor Yago, por aun estar hasta arriba de trabajo confiar en mí y darme las directrices necesarias para realizar este proyecto.

Y por último y no menos importante a todos esos profesores que supieron transmitir el entusiasmo que ellos sentían por su campo y emocionarnos con ellos.

A todos ellos, gracias.

Tabla de contenido

Índice de tablas

Índice de Ilustraciones

1. Introducción y objetivos

1.1. Introducción

La inteligencia artificial es un campo muy importante a lo que programación de videojuegos se refiere. Ya que da la posibilidad de tener distintas opciones independientes completamente del jugador. Ya sea como rival, aliado o simplemente un elemento para interactuar.

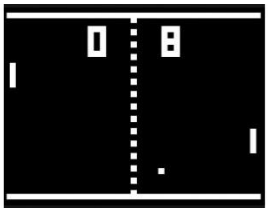


Ilustración 1: Juego Pong

A lo largo de los años esta inteligencia artificial se ha ido haciendo más compleja, pasando de describir simplemente un comportamiento como en el *Pong* (1972)[1], en el cual solo estaban definidas las acciones del contrincante a través de unas pocas reglas. Hasta juegos como *Dragon Age: Inquisition* (2014)[2] en el que la IA está presente en aliados, enemigos y otros personajes neutrales, los cuales según las acciones del jugador responden o actúan de una manera u otra.

Ilustración 2: Juego Dragon Age: Inquisition

Para poder centrarnos en el desarrollo de la inteligencia artificial utilizaremos el videojuego realizado por Marco A. Pajares: *Paintball UC3M* (2013) [3], el cual consiste en un jugador con un arma que dispara pintura tiene que acabar con sus rivales, y la única IA presente es la de los enemigos, que simplemente consiste en cuanto ven al jugador lo persiguen y le disparan.

En este proyecto veremos los distintos usos dados a la inteligencia artificial en otros videojuegos y los cuales se elegirán para este caso en concreto.

1.2. Motivación y objetivos

La motivación que me impulsó a seleccionar este proyecto es porque desde pequeño siempre me fascinó la robótica, me encantaba montar robots y verlos desplazarse, era todo un reto como de la nada podía salir algo que se moviera. Según fui creciendo seguí investigando y no quería conformarme con que solo anduvieran. Asique me metí en cursos de programación, lo cual me atrapo completamente, el hecho de como de nada, poder crear tanto. Más adelante,

descubrí que los videojuegos, uno de mis mayores hobbies, estaban íntimamente relacionados con la programación. Y no solo eso, había un campo que se ocupaba de los comportamientos, el cual me recordó cuando quería algo más de los robots, este era el de la inteligencia artificial. Con este proyecto quiero sumergirme de lleno en este campo y explotar al máximo sus posibilidades, algo que llevo deseando bastante tiempo y ahora tengo la oportunidad de llevarlo a cabo.

El objetivo de este proyecto es desarrollar una capa de inteligencia artificial básica para el comportamiento de los NPC (non-player characters) del videojuego *Paintball UC3M* (2013). Esta capa de inteligencia artificial pretende ser una prueba de concepto que permita dar una idea sobre las distintas posibilidades que puede aportar la inteligencia artificial en cuanto a dinamismo y originalidad a este videojuego. El objetivo principal es desarrollar personalidades para NPCs ya sean adversarios o simplemente personas con las que poder interactuar. Para ello será necesario en primer lugar analizar las técnicas existentes, elaborar un diseño y modelado adecuados, desarrollar dicha capa para finalmente integrarla en el código existente y probar su funcionalidad.

1.3. Estructura de la memoria

Este documento está dividido en siete partes, los cuales son especificados y descritos a continuación:

- **Planteamiento del problema**
  - Se tratará desde el estado del arte de la inteligencia artificial, los videojuegos y la relación entre ellos. Además de los requisitos que tendrá que resolver la solución.
- **Diseño de solución técnica**
  - Se analizarán las distintas alternativas para solucionar el problema, evaluadas según las ventajas e inconvenientes de cada una. Y para finalizar, una descripción detallada del desarrollo de la seleccionada.
- **Pruebas y evaluación**
  - Se explicará el método de evaluación a utilizar, se analizarán cada una de las pruebas realizadas y se realizara un estudio de las posibles mejoras a todos los problemas encontrados.
- **Aspectos económicos y legales**
  - Se hará referencia a los presupuestos estimados para el proyecto y los temas legales a tener en cuenta para su realización.
- **Planificación de trabajo**
  - Se desarrollará un planteamiento inicial y un diagrama de Gantt. A lo largo del proyecto se ira indicando los tiempos reales que ha ido costando en realidad cada tarea y al final se realizará una comparativa, haciendo hincapié del porque no se ha cumplido las expectativas y como solucionarlo para la próxima vez.
- **Conclusiones**
  - Al finalizar el proyecto se analizará si se han cumplido los objetivos, que cambios se han tenido que realizar, el porqué de esos cambios, las líneas futuras de trabajo y una conclusión final sobre el proyecto.

2. Planteamiento del problema

2.1. Introducción

El problema que se va tratar en este proyecto es la incorporación de varios NPC, con distintas inteligencias artificiales, que a partir de ahora se les llamara personalidades, en un videojuego ya creado.

El cometido de estos NPC es otorgar diversidad y dinamismo, ya sea como persona que camina por el entorno que protesta si le dispara, o como enemigo que se esconde y busca el momento adecuado para disparar. Estos elementos harán mucho más atractivo y divertido el videojuego.

2.2. Análisis del estado del arte

El tema de la inteligencia artificial en los videojuegos es relativamente moderno, sobre todo si se considera desde que es llamado así. A continuación, se realizara un estudio de ambos campos por separado y para concluir, la historia relacionada entre ambos.

2.2.1. Inteligencia artificial

Desde la antigüedad el ser humano ha planteado la imitación de "inteligencia" en seres artificiales [4] como en el mito griego de Galatea, en que los dioses dotaban vida a una escultura, o la novela de Frankenstein, un científico crea un cuerpo con restos humanos y le da vida. Estos son ejemplos desde un planteamiento literario, en los cuales se puede observar la idea de que algo artificial pueda conseguir inteligencia. Se podría considerar que así empezó la IA, con ideas fantásticas que alguien pensó que se podían llevar acabo.

Ilustración 3: Frankenstein

A lo largo de la historia se han creado mecanismos que podían simular acciones que requerían "inteligencia" y autómatas como el pato mecánico de Jacques Vancanson en 1738 capaz de nadar, batir las alas, comer y expulsar excrementos [5].

Pero solo a partir de 1950 podemos hablar de inteligencia artificial moderna debido a que Alan Turing público "Computing Machinery and Intelligence". En el que indicaba la posibilidad de que una máquina pueda imitar el comportamiento de la mente humana. De aquí salió la cuestión de si las maquinas podían pensar. Debido al dilema del significado "pensar", Turing creo un test para averiguarlo de manera experimental. La prueba consiste en que un humano realiza preguntas a diferentes voluntarios, y tiene que indicar si quien le responde es una maquina o un humano. Si una maquina es capaz de hacerse pasar por humano es que posee "inteligencia".

Los años posteriores al artículo de Turing, la IA gozo de gran popularidad y se crearon multitud de proyectos. Entre ellos *ELIZA*, un programa de charla, o algoritmos de búsqueda de estados. Pero esta prosperidad duro poco debido a que los ordenadores de la época estaban muy limitados, ya fueran por cómputo o almacenamiento. Además se encontraba la paradoja de Moravec, que se da debido a que el razonamiento es más fácil de computar que las capacidades sensoriales y motoras [6].

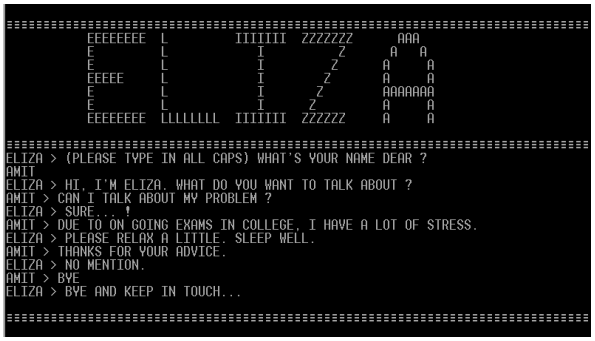


Ilustración 4: Programa de charla ELIZA

En los ochenta surgieron los llamados sistemas expertos. Los cuales consisten en un programa con una base de conocimientos obtenidos de uno o varios expertos, de ahí su nombre, sobre un área determinada y a través de esta información resolver problemas específicos. Estos sistemas también se encontraron con problemas como costes muy elevados, no aprendían con el tiempo y podían cometer errores muy graves debido a las áreas en los que se aplicaban.

Más adelante se publica en el campo de redes de neuronas sobre el perceptron multicapa [7]. El cual podía abordar problemas no lineales, como la resolución del operador lógico XOR, y además que se demostró que era un aproximador universal. Es decir podía aproximar cualquier función a una más simple.

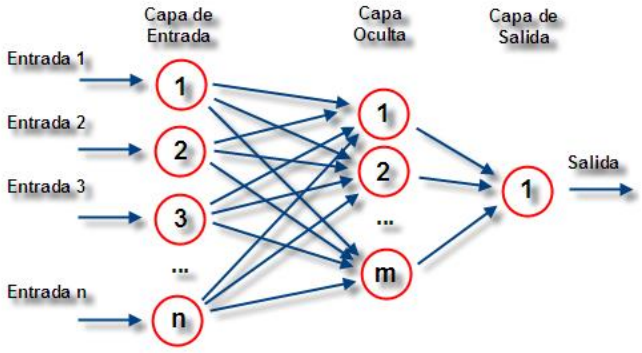


Ilustración 5: Perceptron Multicapa

A día de hoy la inteligencia artificial es capaz de darle la capacidad a un niño robot de que aprenda por sí mismo a tocar un instrumento [8]. Ya queda lejos la complicación de los antiguos ordenadores que les faltaba la capacidad suficiente, aunque en problemas concretos como la simulación del cerebro humano estamos lejos de ese potencial, y podemos afrontar problema que antaño eran impensables.

2.2.2. Videojuegos

Los orígenes de los videojuegos no están tan claros debido a falta de documentación, que no mostraba gráficos por una pantalla u otros que fueron considerados simplemente juegos electrónicos. Pero si caben destacar dos videojuegos que marcaron un inicio: *OXO*, que era el clásico de las tres en raya, maquina contra jugador, en el que el mando era un dial de teléfono; y *tennis for two*, que con un osciloscopio creo un simulador de tenis de mesa para dos jugadores. Este último fue la inspiración para uno de los primeros videojuegos comerciales, el *Pong* [9].

A partir de 1970, empieza el boom de los videojuegos comerciales que caben destacar *Asteroids*, *Space Invaders* y el ya mencionado *Pong*. Esto permito que con el tiempo crecieran salas recreativas, con juegos como *Pacman*, y el desarrolo de consolas domesticas como la Atari 5200, la Commodore 64 y el mítico Spectrum.

Ilustración 6: Juego Space Invaders

El mundo de los videojuegos siguió evolucionando con consolas como la NES de la empresa Nintendo y llegando a ordenadores personales, es decir a una plataforma no creada para videojuegos. Apareció el adictivo *Tetris* y el clásico *Super Mario Bross*, el cual marco un antes y un después porque fue el primer juego que tenía un principio y un fin, a diferencia de sus predecesores que se repetía una y otra vez el mismo nivel aumentando la dificultad.

Una de las mayores innovaciones, en ese tiempo, fue la creación de consolas portátiles, el cual permitía jugar en cualquier lugar, expandiendo y haciendo más visible a los videojuegos. Y la primera, en este campo, fue Game & Watch de Nintendo, una maquina con tecnología LCD que incorporaba algunos juegos. Pero la más conocida y la que marcó un antes y un después fue, su sucesora, la Game Boy, por su inmenso catálogo de videojuegos y por sus revisiones, las cuales fueron de hacerla más ligera hasta poder jugar a color.

Ilustración 7: Game boy

En los noventa, llego el formato CD-ROM, la tecnología que hizo tan popular y dinámica a la PlayStation de Sony, y los juegos en tres dimensiones. También fue el mayor aumento de jugadores, debido a la gran venta de consolas y sobretodo de ordenadores domésticos. Los juegos que marcaron esta época fueron muchos pero caben destacar: *Doom*, *Donkey Kong Country* y *Killer Instinct*.

Al inicio del siglo XXI, los videojuegos se convertían en un negocio muy rentable. Con solidando a tres empresas, Sony, Microsoft y Nintendo, como líderes del sector de videoconsolas. Sony sacando PlayStation 2, Nintendo con su nueva portátil Game Boy Advance y su sobre mesa Game Cube y Microsoft con Xbox. El PC, apoyado por Microsoft, seguía siendo popular para jugar a videojuegos pero era la alternativa más cara. Esta guerra continúo con Nintendo sacando una nueva portátil innovadora con dos pantallas siendo una de ellas táctil, Nintendo DS, y Sony contratacando con su propia portátil que apostaba por una mayor potencia, PSP. Y en videojuegos, gracias al nuevo potencial, se crearon nuevos géneros como el mundo abierto, y juegos más pausados como el RPG (Rol Playing Game) por turnos o las plataformas dio paso a juegos más de acción como *Kingdom Hearts o Prince of Persia: Las Arenas del Tiempo*.

Ilustración 8: Juego Kingdom Hearts

La siguiente generación de videoconsolas se caracterizó por los juegos online, los cuales principalmente se encontraban en PC. Y fueron Xbox 360 por Microsoft, Wii de Nintendo y PlayStation 3 de Sony. Las de Microsoft y Sony se especializaron y una potencia grafica muy significativa, mientras que Nintendo aposto por una nueva manera de jugar, un mando que se agarraba como el mango de una espada que podía capturar los movimientos para ser utilizados en el juego. Y en el apartado de videojuegos, todavía salen alguno para estas consolas, los más vendidos fueron los que más aprovechaban el juego en línea y la potencia gráfica, en lo cual destaco el género Shotter (de disparos), con juegos como *Call of Duty 4: Modern Warfare* o *Battlefield: Bad Company* y los deportivos como el *FIFA 09* o *Pro Evolution Soccer 2011*. En innovación destacaron juegos como *Mirror's Edge*, *Assassin's Creed* o *The Elder Scrolls IV: Oblivion*. Con esta generación la gente que jugaba en PC disminuyo considerablemente y se especializo sobretodo en el género MMORPG (Massive Multiplayer Online Rol Playing Game) con juegos como el *World of Warcraft* o *Lineage II*.

Ilustración 9: Juego Lineage II

A día de hoy el panorama es el siguiente: en portátiles Nintendo posee la 3DS, una consola capaz de simular el efecto 3D sin gafas, y Sony la PsVita, con una pantalla delantera táctil y atrás un panel también táctil y con una gran potencia gráfica. En sobremesa Nintendo con WiiU, Sony con PlayStation 4 y Microsoft con Xbox One. Estas consolas apuestan por las redes sociales y crear una comunidad. Y a lo que videojuegos se refiere no se puede tratar mucho ya que las consolas son muy recientes. En PC surgen plataformas como Steam o GOG con juegos increíblemente baratos debido a esto y a que las consolas están siendo tan cara esta plataforma se convierte en la más versátil pudiendo ser de las más baratas permitiendo jugar a una gran multitud de juegos, a de las más caras y potentes. Esto ha generado un gran aumento de jugadores en PC.

Ilustración 10: Logo de Steam

2.2.3. Uso de la inteligencia artificial en videojuegos

La inteligencia artificial siempre ha estado muy influenciada por los videojuegos y viceversa. Podríamos definir como punto de partida el año 1959 cuando Arthur Samuel publicó su libro “Some studies in machine learning using the game of checkers” con el cual creo un juego de damas con un sistema de inteligencia artificial con autoaprendizaje [10].

Ilustración 11: Juego Qwak

En los años 70 surgieron juegos para un solo jugador, tales como *Qwak*, *Pursuit* o *Star Trek*, en los cuales el movimiento de los enemigos estaba almacenado. En esta década también surgieron los microprocesadores lo que permitió a los videojuegos agregar aleatoriedad a los movimientos. A finales, surgieron juegos como *Space Invaders* y *Galaxian* que utilizaron funciones hash basadas en las acciones del jugador para adaptarse a ellas y aumentar así la dificultad [11].

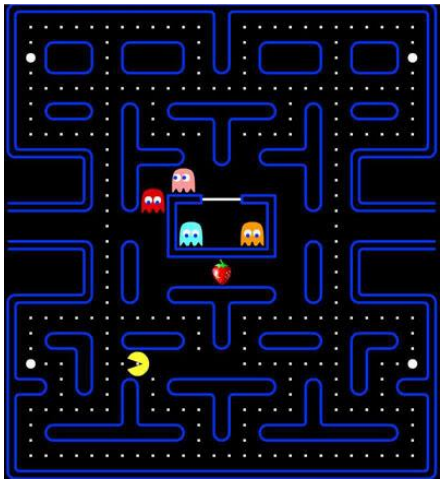


Ilustración 12: Juego Pac-man

La siguiente década se caracterizó por la creación de "personalidad" a los enemigos, es decir comportamientos característicos distintos entre NPC, los juegos más representativos de esta característica y época fueron *Pac-Man*, cada fantasma tomaba diferentes decisiones para llegar a Pac-Man, *Karate Champ*, cada rival tenia diferentes características, y *Madden Football*, en este último lo que se intento fue reproducir el comportamiento de los jugadores profesionales que se representaban en el juego [12].

Ilustración 13: Juego Karate champ

Con el Boom del mercado de los videojuegos en los 90 surgieron consolas muy potentes lo que permitió la utilización de nuevas técnicas de IA. Tales como máquina de estados, el personaje va cambiando las acciones que realiza según el estado en el que se encuentre, el cual va cambiando según lo que va detectando, ya sean acciones del jugador o cambio de estados de otros enemigos; búsqueda de

caminos, el cual tiene una serie de comportamientos y analiza cual son los que tiene que hacer; decisiones en tiempo real, el NPC decide en cada momento lo que va a realizar; redes de neuronas, es un sistema el cual según una serie de entradas dará una salida, en nuestro caso, la entrada podría ser el entorno y la posición del jugador, entonces la salida nos daría la acción a realizar [13].

Ilustración 14: Juego Far Cry

El comienzo del siglo XXI se caracterizó por utilizar estas técnicas de manera innovadora, por ejemplo, en *Halo* los enemigos reconocían amenazas y reaccionaban ante ellas, en *Far Cry* analizaban el estilo de juego del jugador y planteaban estrategias militares en consecuencia, en *Left 4 Dead* se creaban evento de manera procedural para cada vez que se jugara de nuevo fueran ocurriendo cosas distintas [14].

Ilustración 15: Juego Shadow of Mordor

A día de hoy se siguen utilizando estas técnicas, pero gracias a la potencia de las nuevas consolas, son llevadas a otro nivel. Como en el caso de *Shadow of Mordor* el cual plantea su sistema génesis el cual consiste que en cuanto matas a un enemigo el resto pueden huir o algún otro ocupar su posición, si alguno escapa puede volver a por ti con una tropa en busca de venganza, cada decisión cuenta para cambiar el mundo, el sistema y decisiones del enemigo.

2.3. Requisitos

Los requisitos permitirán definir las pautas que tendrá que cumplir la IA tanto funcionales como no funcionales.

A continuación especificaremos la descripción de la tabla de requisitos y los distintos campos que la componen para poder incluir posteriormente la enumeración de todos ellos.

Los requisitos vendrán definidos en una tabla como la siguiente:

Identificador	
Prioridad:	
Necesidad:	
Claridad:	
Verificabilidad:	
Estabilidad:	
Descripción:	

Tabla 1: Plantilla de requisitos

A continuación se van a explicar en detalle cada uno de los campos de la tabla:

- **Identificador**: Este campo sirve para distinguir un requisito de otro inequívocamente. Cada identificador seguirá la siguiente nomenclatura:
  - **XXX-nnn**:
    - **XXX**: Tipo de requisito:
      - **FNC**: Funcional.
      - **NFN**: No Funcional.
    - **nnn**: Indica el número del requisito que ira desde el 001 hasta el 999
- **Prioridad**: Este campo indica el grado de prioridad con el que debe ser resuelto un requisito. Podrá tener uno de los siguientes valores:
  - **Alta**: El diseño será de carácter prioritario.
  - **Media**: El diseño será de carácter moderado.
  - **Baja**: El diseño será de carácter inferior.
- **Necesidad**: Indica la necesidad de incorporar el requisito en el sistema. Los valores que puede tener son:
  - **Esencial**: El requisito debe incorporarse al sistema.
  - **Deseable**: El requisito debe incorporarse al sistema siempre y cuando no cause problemas ni conflictos con otros.
  - **Opcional**: El requisito solo se incorporara en caso en el que se esté cumpliendo la planificación y no cause problemas ni conflictos con otros.
- **Claridad**: Este campo indica si el requisito esta expresado correctamente. Podrá tener uno de los siguientes valores:
  - **Alta**: El requisito solo puede tener una interpretación.
  - **Media**: El requisito está bien definido y es improbable que surjan dudas.
  - **Baja**: El requisito puede tener varias interpretaciones dependiendo del contexto.
- **Verificabilidad**: Indica la posibilidad de comprobar que el requisito se haya incorporado al sistema. Los valores que puede tener son:

- **Alta**: Se puede comprobar de manera sencilla.
- **Media**: La comprobación puede llevar algo de tiempo.
- **Baja**: Es prácticamente imposible comprobar si el requisito se ha incluido al sistema.

- **Estabilidad**: Define la posibilidad de que un requisito no se modifique durante el desarrollo del proyecto.
- **Descripción**: En este campo se incluye una descripción del requisito.

2.3.1. Requisitos funcionales

En este apartado vamos a definir los requisitos que especifican el propósito del software. Los cuales son los siguientes:

FNC-001	
Prioridad:	Alta
Necesidad:	Esencial
Claridad:	Alta
Verificabilidad:	Alta
Estabilidad:	Durante toda la vida del sistema
Descripción:	Interacción con el jugador

Tabla 2: FNC-001

FNC-002	
Prioridad:	Alta
Necesidad:	Esencial
Claridad:	Alta
Verificabilidad:	Alta
Estabilidad:	Durante toda la vida del sistema
Descripción:	Como enemigo, perseguir al jugador

Tabla 3: FNC-002

FNC-003	
Prioridad:	Alta
Necesidad:	Esencial
Claridad:	Alta
Verificabilidad:	Alta
Estabilidad:	Durante toda la vida del sistema
Descripción:	Como enemigo, disparar al jugador

Tabla 4: FNC-003

FNC-004	
Prioridad:	Alta
Necesidad:	Esencial
Claridad:	Alta
Verificabilidad:	Alta
Estabilidad:	Durante toda la vida del sistema
Descripción:	Como enemigo, esconderse del jugador

Tabla 5: FNC-004

FNC-005	
Prioridad:	Alta
Necesidad:	Esencial

Claridad:	Alta
Verificabilidad:	Alta
Estabilidad:	Durante toda la vida del sistema
Descripción:	Como enemigo, no entorpecer a sus compañeros

Tabla 6: FNC-005

FNC-006	
Prioridad:	Alta
Necesidad:	Esencial
Claridad:	Alta
Verificabilidad:	Alta
Estabilidad:	Durante toda la vida del sistema
Descripción:	Como NPC, desplazarse siguiendo una ruta

Tabla 7: FNC-006

FNC-007	
Prioridad:	Alta
Necesidad:	Esencial
Claridad:	Alta
Verificabilidad:	Alta
Estabilidad:	Durante toda la vida del sistema
Descripción:	Como NPC, buscar refugio cuando llueve

Tabla 8: FNC-007

FNC-008	
Prioridad:	Alta
Necesidad:	Esencial
Claridad:	Alta
Verificabilidad:	Alta
Estabilidad:	Durante toda la vida del sistema
Descripción:	Como NPC, ser capaz de chocar con el jugador

Tabla 9: FNC-008

FNC-009	
Prioridad:	Alta
Necesidad:	Esencial
Claridad:	Media
Verificabilidad:	Alta
Estabilidad:	Durante toda la vida del sistema
Descripción:	Como NPC, tras un choque mostrar emociones

Tabla 10: FNC-009

FNC-010	
Prioridad:	Alta
Necesidad:	Esencial
Claridad:	Alta
Verificabilidad:	Alta
Estabilidad:	Durante toda la vida del sistema
Descripción:	Como NPC, si es disparado por el jugador, perseguirlo y golpearlo

Tabla 11: FNC-010

FNC-011	
Prioridad:	Alta
Necesidad:	Esencial
Claridad:	Alta
Verificabilidad:	Alta
Estabilidad:	Durante toda la vida del sistema
Descripción:	Como NPC, dejar de perseguir al jugador después de 3 segundos

Tabla 12: FNC-011

FNC-012	
Prioridad:	Alta
Necesidad:	Esencial
Claridad:	Alta
Verificabilidad:	Alta
Estabilidad:	Durante toda la vida del sistema
Descripción:	Como NPC, si es disparado por el jugador, salir huyendo

Tabla 13: FNC-012

FNC-013	
Prioridad:	Alta
Necesidad:	Esencial
Claridad:	Alta
Verificabilidad:	Alta
Estabilidad:	Durante toda la vida del sistema
Descripción:	Como NPC, dejar de huir después de 3 segundos

Tabla 14: FNC-13

FNC-014	
Prioridad:	Alta
Necesidad:	Esencial
Claridad:	Alta
Verificabilidad:	Alta
Estabilidad:	Durante toda la vida del sistema
Descripción:	El número de NPCs al comenzar la partida se definirá de manera aleatoria entre 6 y 18

Tabla 15: FNC-014

FNC-015	
Prioridad:	Alta
Necesidad:	Esencial
Claridad:	Alta
Verificabilidad:	Alta
Estabilidad:	Durante toda la vida del sistema
Descripción:	Según llegue la noche los NPCs deberán irse dirigiendo a las afueras de la universidad

Tabla 16: FNC-015

FNC-016	
Prioridad:	Alta
Necesidad:	Opcional
Claridad:	Alta
Verificabilidad:	Alta

Estabilidad:	Durante toda la vida del sistema
Descripción:	Los NPC interactúan entre ellos

Tabla 17: FNC-016

2.3.2. Requisitos no funcionales

En este apartado vamos a definir los requisitos que especifican el modo de realizar las tareas y las limitaciones de las mismas. Los cuales son los siguientes:

NFN-001	
Prioridad:	Alta
Necesidad:	Esencial
Claridad:	Alta
Verificabilidad:	Alta
Estabilidad:	Durante toda la vida del sistema
Descripción:	Seis modelos diferentes de NPC

Tabla 18: NFN-001

NFN-002	
Prioridad:	Alta
Necesidad:	Esencial
Claridad:	Media
Verificabilidad:	Alta
Estabilidad:	Durante toda la vida del sistema
Descripción:	Tiempo de reacción de los NPCs menor de un segundo

Tabla 19: NFN-002

NFN-003	
Prioridad:	Alta
Necesidad:	Esencial
Claridad:	Alta
Verificabilidad:	Alta
Estabilidad:	Durante toda la vida del sistema
Descripción:	El comportamiento del NPC no tiene que interferir con su modelado

Tabla 20: NFN-003

3. Diseño de la solución técnica

3.1. Introducción

En este apartado se va analizar las posibles soluciones para la generación de la inteligencia artificial para los NPC y enemigos del juego. Se sopesara los pros y los contras de cada alternativa. Una vez elegida se explicara su desarrollo de manera detallada.

3.2. Diseño de la solución inicial

Para diseñar una solución tenemos que tener en cuenta que vamos a trabajar con un proyecto de terceros, es decir, tendremos unos límites muy definidos como:

- **Motor del juego**: Tendremos que utilizar el UDK 3 debido a que el juego, en el que vamos a incorporar nuestro sistema, ha sido desarrollado en el mismo. Y como la versión 3 del motor no es compatible con la más nueva, la versión 4, ya que no utilizan el mismo lenguaje de programación.
- **Comprensión**: Como trabajamos con otro proyecto una parte del tiempo ira al análisis y comprensión del mismo, para entender cómo funciona y se realizan las cosas para una mayor y mejor integración.

Además de estos límites surgen otros más comunes como:

- **Aprendizaje**: Parte del tiempo se invertirá en aprender en manejar algunos de los distintos programas necesarios para el desarrollo del sistema.
- **Equilibrio**: La incorporación de la IA debe ser un añadido positivo y no perjudicar al estilo, ni a la diversión del juego.
- **Fecha de entrega**: La fecha límite para finalizar el proyecto está entre el 23 y 27 de septiembre de 2015.

Además de las limitaciones valoramos otras ideas para poder apoyar en ellas y coger lo mejor de cada una para mejorar nuestra solución y ahorrar en tiempo.

Ilustración 16: Juego Grand Theft Auto V

- ***Grand Theft Auto***: Uno de los principales modelos que vamos a tener en cuenta. El cual también fue referente en el proyecto que estamos utilizando como base. *Grand Thef Auto* es una saga de videojuegos de mundo abierto en el cual el jugador no tiene un camino predefinido que seguir. Esta saga también se caracteriza por la interacción

del jugador con los transeúntes de la ciudad (NPCs), ya que puede embarcarse en peleas o que la gente al verte con un arma huya. Este comportamiento es un buen ejemplo de lo que queremos llegar alcanzar con nuestro sistema asique estos juegos serán un modelo a seguir en el desarrollo.

Ilustración 17: Juego Assassin's Creed Unity

- ***Assassin's Creed***: En esta saga, también de mundo abierto aunque algo más limitado que el anterior, los NPCs también reaccionan a las acciones del jugador, como por ejemplo si matas a alguien y te ven empiezan a gritar y avisan a la guardia, si el jugador va corriendo puede chocar con uno y ambos pueden caer al suelo o tropezar, mientras que si va andando puede apartar a los NPCs o esquivarlos. De estos juegos vamos a tomar como referencia la interacción jugador-NPC ya que es una de las mejores y un buen modelo a seguir para nuestro proyecto.

Para finalizar, detallar que al ser nuestra inteligencia artificial tan básica buscaremos técnicas clásicas y sencillas para su desarrollo.

3.3. Desarrollo de alternativas

Una vez tenemos los conceptos básicos para crear un diseño, mencionados en el apartado anterior, pasamos a analizar las posibles alternativas para el desarrollo. Las técnicas clásicas más comunes son:

3.3.1 Búsqueda de caminos

A través de un grafo se representa las distintas posiciones a seguir para llegar a una posición final. Se suele utilizar para esquivar obstáculos, posicionarse para mejor ángulo de tiro o para ocultarse.

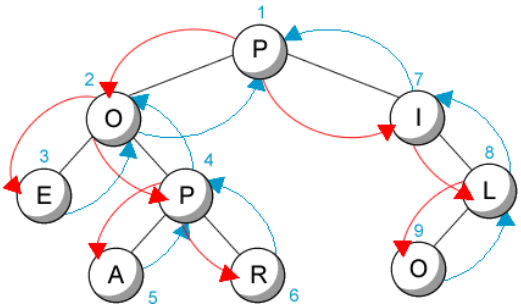


Ilustración 18: Búsqueda en profundidad

- **Búsqueda en profundidad**: recorremos el grafo por el primogénito de cada nodo que llegamos, hasta llegar al nodo que estamos buscando o hasta el final. En el caso de no poder seguir recorriendo esa rama volveríamos al nodo anterior, recorreríamos a través del siguiente hijo y así hasta encontrar la posición final o recorrer todo el árbol, en tal caso no sería posible llegar a la posición deseada.

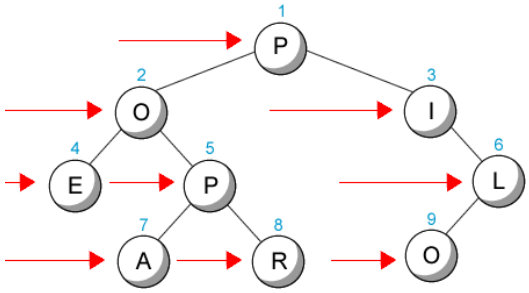


Ilustración 19: Búsqueda en amplitud

- **Búsqueda en amplitud**: cada vez que llegamos a un nodo recorremos todos sus hijos, si ninguno de ellos es la posición final seguimos por el primer hijo. Si no podemos seguir recorriendo la rama retrocedemos un nodo y seguiríamos por el siguiente hijo. Si en el árbol no se encontrara el nodo final esto indicara que no es posible llegar a esa posición.
- **Búsqueda con heurísticas**: cada vez que llegamos a un nodo a la hora de decidir cuál será el siguiente nodo en recorrer se hace en base a una heurística.

3.3.2 Máquina de estados

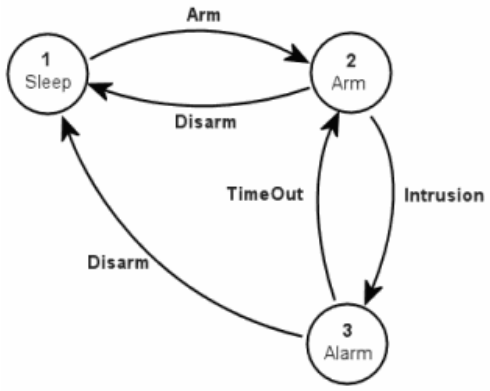


Ilustración 20: Máquina de estados

A través de un autómata determinista finito siendo cada nodo un estado, como vigilia, disparando o persiguiendo, y las transiciones son condiciones que se tienen que dar para que la maquina cambie entre ellos. Es decir el NPC está en un estado y cuando se dé una de las condiciones que salen de él cambiara al otro estado que se encuentra al otro extremo.

3.3.3 Aprendizaje por refuerzo

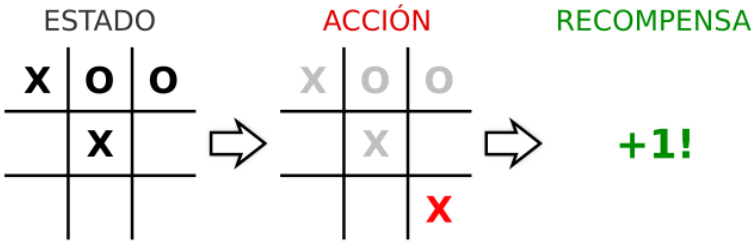


Ilustración 21: Aprendizaje por refuerzo

Se simula al NPC en diferentes situaciones a las cuales se le indicara cuanto de buena es la acción que ha tomado, de esta manera se genera un modelo para cuando lo situemos en el juego sea capaz de predecir qué hacer en cada momento gracias a las experiencias previas. Por lo que requerirá un periodo de aprendizaje para cada personalidad.

3.4. Ventaja e inconvenientes de los diseños

A continuación estudiaremos los pros y los contras de cada una de los tipos de desarrollo mencionados anteriormente para el diseño de nuestra inteligencia artificial.

3.4.1 Búsqueda de caminos

- **Ventajas**
  - Fácil implementación.
  - Buenos resultados para la selección de posiciones cercanas.
  - Mucha documentación al respecto.
- **Inconvenientes**
  - Crecimiento de la complejidad de manera exponencial, es decir mucho más tiempo de cómputo, según el tamaño del grafo.
  - Difícil incorporación de acciones en el grafo.
  - No tengo experiencia previa en su uso.

3.4.2 Máquina de estados

- **Ventajas**
  - Muy fácil implementación.
  - Motor de juego con opciones específicas para este diseño.
  - Mucha documentación al respecto, además de específica del motor a usar.
  - Muy versátil, ya sea para desplazamientos o acciones.
  - Bajo coste computacional.
  - Tengo experiencia previa en su uso.
- **Inconvenientes**
  - Necesitan mucha precisión a la hora de su definición ya que es fácil caer en la redundancia o crear circuitos cerrados.

3.4.3 Aprendizaje por refuerzo

- **Ventajas**
  - La inteligencia artificial “aprende” sola a actuar.

- Más naturalidad a la hora de tomar decisiones.
- Puede llegar a opciones que ni si quiera nos hubiéramos planteado.
- Muy versátil, ya sea para desplazamientos o acciones.

- **Inconvenientes**
  - Alto coste computacional.
  - Mucho tiempo hasta que se llega a una inteligencia artificial útil.
  - Poca documentación sobre este diseño aplicado a los videojuegos.
  - No tengo experiencia en su uso.

3.4.4 Conclusiones

Si analizamos cada una de las opciones nos damos cuenta que la que mejor se adapta a nuestro proyecto es la máquina de estados. Ya que tenemos muchas facilidades lo cual nos permitirá alcanzar nuestros objetivos lo antes posible y así poder realizar más pruebas y mejoras. También la versatilidad es una de sus principales características ya que nos permitirá hacer parecer a la IA más compleja. Y al ser la de menor coste computacional nos aseguraremos que el rendimiento del videojuego apenas decaiga.

El aprendizaje por refuerzo es también una gran alternativa pero debido al desconocimiento de cuánto podría tardar en obtener una inteligencia artificial suficientemente útil la convierte en una apuesta arriesgada.

3.5. Desarrollo del diseño elegido

Para comenzar a desarrollar nuestra solución lo primero es obtener los seis modelos distintos para nuestros NPCs. Para ello buscamos modelos en las webs: http://www.cgtrader.com, http://tf3dm.com y http://www.turbosquid.com.

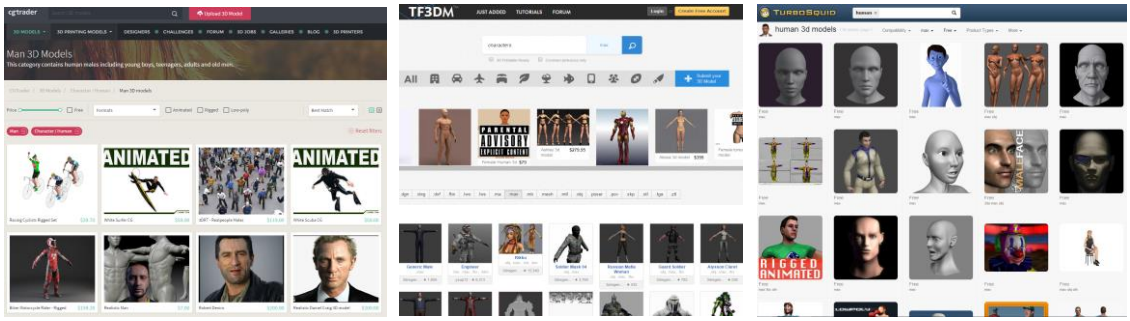


Ilustración 22: Webs de modelos 3D

En ellas encontramos una gran variedad de modelos pero pocos son los que se adaptan al estilo artístico del juego, aun así seleccionamos algunos para hacer unas pruebas y comprobar que podemos utilizarlo con el motor del juego. El primer problema que hayamos es que no tienen un esqueleto para poderlos animar, esto nos hará perder tiempo, aun así le damos un esqueleto a uno de estos modelos con el programa de 3D Max ya que tenemos experiencia previa en su utilización:

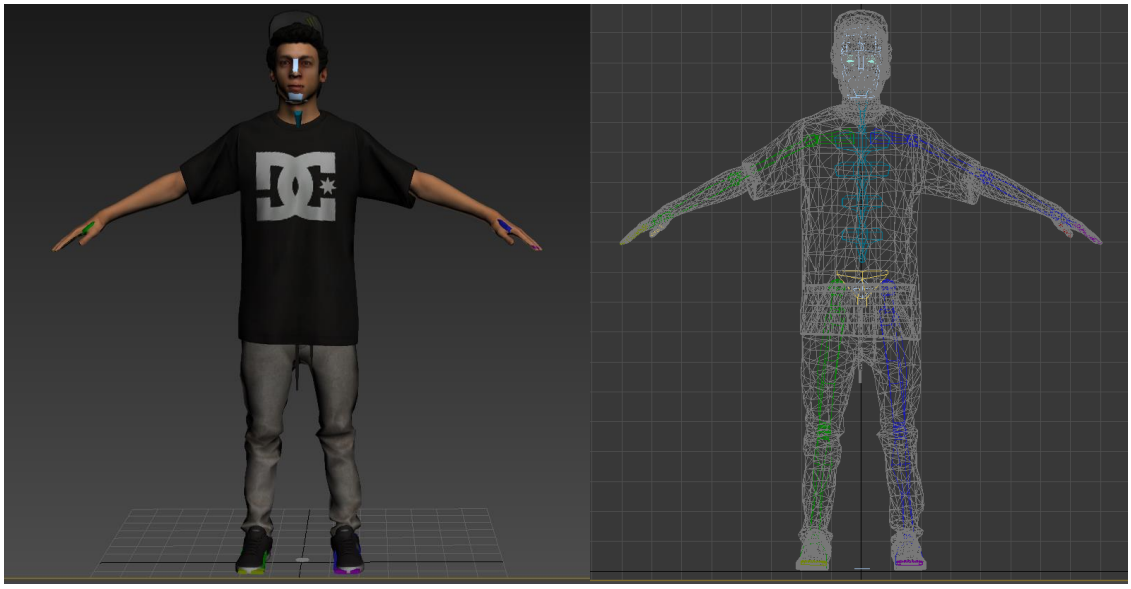


Ilustración 23: Modelo 3D con esqueleto

Después de este trabajo meticuloso, ya que cualquier fallo a la hora de colocar los huesos puede dar a deformaciones a la hora de animar, lo exportamos al motor *UDK* para probar el esqueleto. En esta prueba comprobamos que el esqueleto no es compatible con las animaciones por defecto. Podríamos intentar colocar el esqueleto por defecto del motor a nuestros modelos, lo cual es más complicado ya que es mucho menos editable y flexible a cambios. Pero investigamos y descubrimos que la herramienta que se utilizó en el proyecto, el cual estamos utilizando de base, no ha desaparecido, lo cual al principio fue lo que nos pareció y por eso no lo teníamos como alternativa, sino que lo ha comprado Adobe y que además de crear modelos 3D, les otorga esqueleto y hay una gran variedad de animaciones compatibles con ellos. Así que optamos por crear nuestros modelos con esta aplicación, *Mixamo* [15].

Ilustración 24: Modelo 3D creado con Mixamo

Después de crear el modelo le damos un esqueleto con la aplicación y después buscamos unas animaciones útiles para nuestro juego.

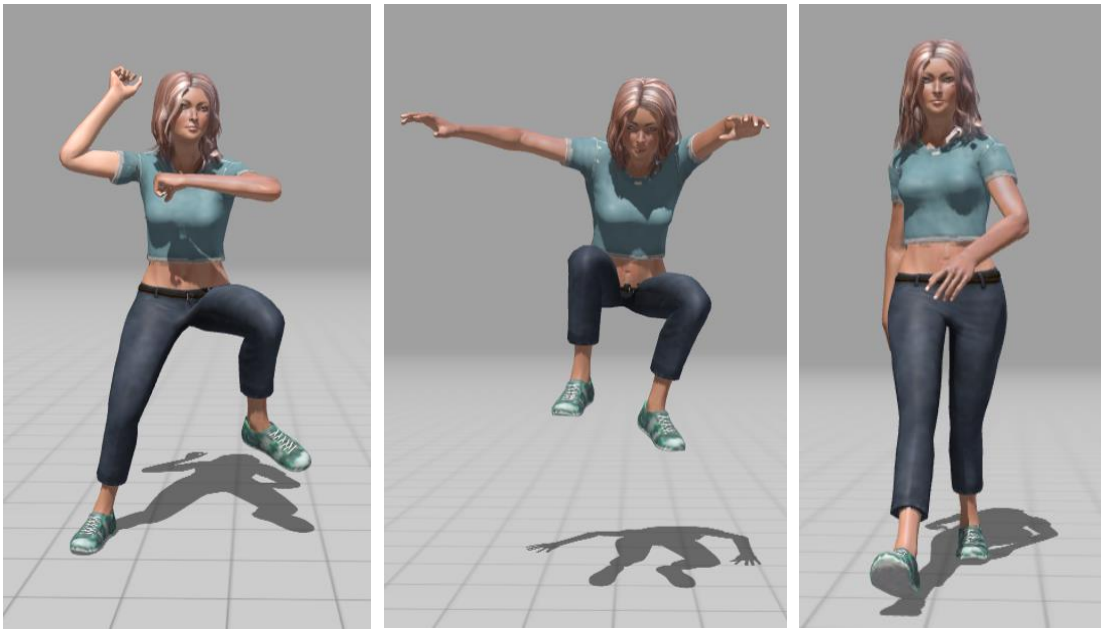


Ilustración 25: Animaciones

Una vez que lo tenemos todo lo exportamos al *UDK* y lo configuramos todo para poderlos utilizar posteriormente desde el código para diseñar nuestra máquina de estados.

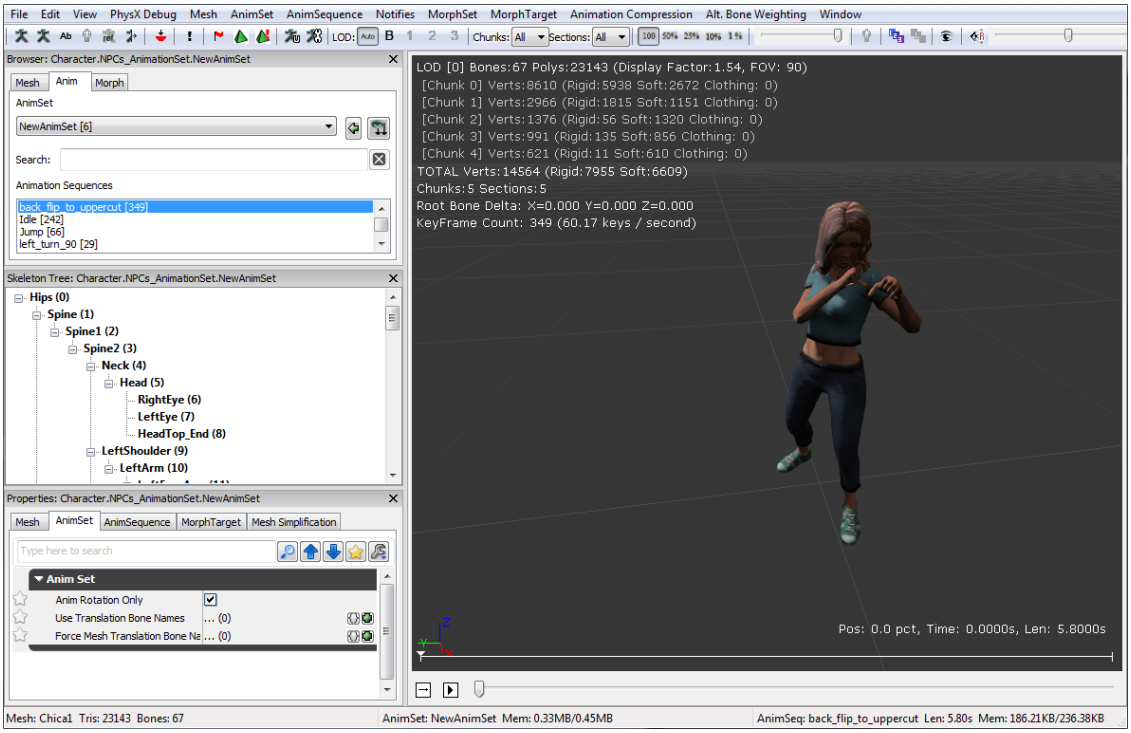


Ilustración 26: Configuración de las animaciones en UDK

Este proceso lo repetimos cada una de las veces que vayamos a crear un nuevo modelo para los NPC. En nuestro caso, cinco veces más para tener un total de seis “personas” distintas andado por la universidad.

Ilustración 27: Modelos de NPCs

Para continuar con nuestro desarrollo analizamos las clases del que dispone el juego, para ello primero creamos un esquema de relaciones y herencias entre ellas:

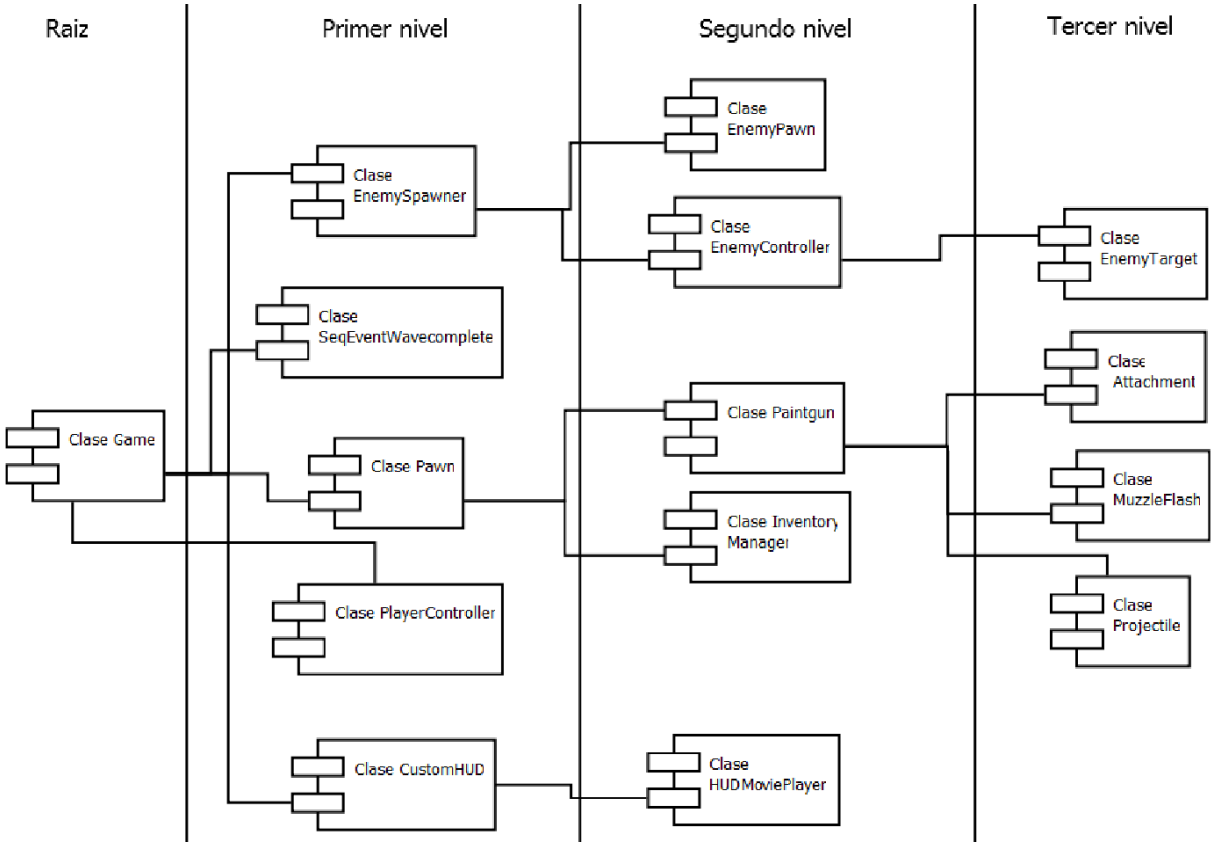


Ilustración 28: Relaciones y herencias entre clases

Una vez que tenemos claro el nivel de herencia y dependencia pasamos analizar de manera individual clase por clase [16]:

Clase Raíz	
Game	Implementa las clases que se van a utilizar en vez de las de por defecto y se encarga de indicar si ha acabado el juego o no

Tabla 21: Clase Raíz

Clases del Primer Nivel	
EnemySpawner	Implementa la creación de enemigos en el juego, dando lugar a oleadas y el numero a generar
SeqEventWavecomplete	Implementa las especificación que indican cuando ha acabado el juego
Pawn	Implementa el control de la cámara, las animaciones del entorno, colisiones y características físicas, tales como la gravedad o velocidad de desplazamiento
PlayerController	Implementa el movimiento del jugador y sus acciones
CustomHUD	Implementa la nueva interfaz creada para el juego

Tabla 22: Clases del Primer Nivel

Clases del Segundo Nivel	
EnemyPawn	Implementa la colocación inicial de enemigos en el entorno
EnemyController	Implementa el comportamiento del enemigo
Paintgun	Implementa el manejo de la pistola de pintura
InventoryManager	Implementa la gestión y almacén de objetos
HUDMoviePlayer	Implementa la animación inicial del juego

Tabla 23: Clases del Segundo Nivel

Clases del Tercer Nivel	
EnemyTarget	Implementa la distancia de acción del enemigo
Attachment	Implementa la posición donde el personaje llevara el arma
MuzzleFlash	Implementa el sistema de partículas de la pistola de pintura
Projectile	Implementa los parámetro del disparo

Tabla 24: Clases del Tercer Nivel

Después de este análisis podemos observar que para el comportamiento del enemigo tendremos que trabajar en la clase ***EnemyController***, ya que es donde está definida el comportamiento básico que tienen ahora mismo [17] [18] [19] [20]. Para ello, además de los dos estados "Mirando", que están buscando al jugador, y "Persiguiendo", en cual persiguen al jugador; vamos a crear el estado "ACubierto", en el que si el jugador les dispara y hay algo para cubrirse lo utilizan, otro que sea "Disparar", el cual cuando este a cierta distancia le disparara. Por lo tanto la máquina de estados quedaría así:

Máquina de estados del Enemigo	
Mirando	Estado en el que busca al jugador
Persiguiendo	Estado en el que va corriendo detrás del jugador
ACubierto	Estado en el que busca un lugar seguro
Disparar	Estado en el que dispara al jugador

Tabla 25: Máquina de estados del Enemigo

Y la transición de estados, es decir, estando en un estado a cual puede cambiar y a cual no tiene acceso:

Transición de estados del Enemigo				
	Mirando	**Persiguiendo**	**ACubierto**	**Disparar**
Mirando	-	X	X	X
Persiguiendo	X	-	X	X
ACubierto	X	-	-	-
Disparar	X	X	-	-

Tabla 26: Transición de estados de Enemigo

Por lo tanto, cuando este en "Mirando" podrá pasar a cualquier estado siempre que se cumpla las condiciones. Cuando este "Persiguiendo", cuando pierda de vista al jugador volverá a "Mirando", cuando le disparen a "ACubierto" y cuando tenga a tiro al jugador a "Disparar". En "ACubierto" al perder al jugador de vista tendrá que estar a la espera hasta que vuelva ponerse a tiro. Y con el "Disparar", se parara para obtener más puntería por lo que le dará al jugador la oportunidad de alejarse, en ese momento pasara a "Persiguiendo" si le sigue viendo o a "Mirando" si lo perdió de vista.

Una parte importante es el cálculo de tiempo y las condiciones exactas, por ejemplo, un problema encontrado es que cada vez que disparaba el jugador todos los enemigos se escondían por lo tanto si el jugador iba constantemente disparando, estos nunca iban detrás del jugador. Por lo tanto, se tuvo que retocar y además de cuando disparara el jugador se puso un tiempo máximo de estar a cubierto y que solo lo hiciera si el jugador se encontraba lejos de él.

Así que para una mayor claridad pasamos a definir de manera detallada cada uno de los estados:

Mirando

- Es el estado inicial, es decir, cuando se posiciona al enemigo en el juego está por defecto en este estado.
- Está a la espera de que se produzca alguna de las condiciones de transición.
- **Transición a “Disparar”**
    - Cuando el jugador este a la vista y a menos de 400 unidades de distancia.
- **Transición a “ACubierto”**
    - Cuando el jugador este a la vista y dispare, siempre y cuando no haya estado más de 5 segundos ya “ACubierto” y que se encuentre a más de 500 unidades de distancia.
- **Transición a “Persiguiendo”**
    - Cuando el jugador se ponga a la vista, el cual UDK tiene una variable que nos informa de ello.

Ilustración 29: Estado Enemigo "Mirando"

Persiguiendo

- Está en desplazamiento detrás del jugador.
- **Transición a “Mirando”**
    - Cuando se pierda de vista al jugador.
- **Transición a “ACubierto”**
    - Cuando el jugador dispare, siempre y cuando no haya estado más de 5 segundos ya “ACubierto” y que se encuentre a más de 500 unidades de distancia.
- **Transición a “Disparar”**
    - Cuando el jugador este a menos de 400 unidades de distancia.

Ilustración 30: Estado Enemigo "Persiguiendo"

ACubierto

- Busca un lugar más cercano para esconderse y espera.
- **Transición a “Mirando”**
    - Cuando el jugador se ponga a la vista o lleve 5 segundos en ese estado.

Ilustración 31: Estado Enemigo "ACubierto"

Disparar

- Se para y se pone a disparar al jugador con una probabilidad del 75% o más, según la distancia, de acertarle.
- **Transición a "Persiguiendo"**
    - Cuando el jugador se aleje más de 400 unidades de distancia y siga a la vista.
- **Transición a "Mirando"**
    - Cuando el jugador se aleje más de 400 unidades de distancia y se pierda de la visión del enemigo.

Ilustración 32: Estado Enemigo "Disparar"

Con esto el comportamiento del enemigo estaría desarrollado y le otorga una mayor diversión al juego a la hora de ir en su búsqueda y "pintarlos" a todos.

Pero todavía nos faltaría el desarrollo de la IA de los NPCs que van a rondar por la universidad para darle mayor naturalidad y vida al entorno. Para ello, como el juego no contemplaba esta posibilidad tendremos que empezar de cero a lo referente al tema. Lo primero sería crear las clases correspondientes: ***NPCSpawner***, ***NPCPawn*** y ***NPCController***. Las cuales quedarían definidas como indica en la siguiente tabla:

Clases de los NPCs	
NPCSpawner	Implementa la creación de los NPCs en el juego, se encarga de decidir el número a generarse
NPCPawn	Implementa la colocación inicial de los NPCs en el entorno
NPCController	Implementa el comportamiento de los NPCs

Tabla 27: Clases de los NPCs

A la hora de desarrollar las clases tenemos que tener en cuenta su homónima a la de los enemigos. En ***NPCSpawner***, no se generaran oleadas y el número de NPCs se decidirá de manera aleatoria. Para ***NPCPawn***, los NPCs no se generaran todos en el mismo punto sino en distinto predefinidos en el editor. Y ***NPCController***, es la que más dista de la de los enemigos ya que define un comportamiento distinto. El cual estará controlado por la siguiente máquina de estados:

Máquina de estados del NPC	
Pasear	Estado en el que el NPC recorrerá la universidad
Golpear	Estado en el que el NPC va hasta el jugador y lo golpea
Huir	Estado en el que el NPC huye en dirección contraria al jugador

Tabla 28: Máquina de estados del NPC

En esta ocasión el estado "Pasear" tendrá distintos comportamientos según la situación, por ejemplo si llueve tomara rumbo a cubrirse y si anochece se dirigirá a las afueras de la universidad, sino recorrerá la universidad de forma normal. Los estados "Golpear" y "Huir" se activaran de la misma manera pero la decisión de la cual se cambiara será de forma aleatoria.

A continuación podemos ver la tabla de transición:

Transición de estados del NPC			
	Pasear	**Golpear**	**Huir**
Pasear	-	X	X
Golpear	X	-	-
Huir	X	-	-

Tabla 29: Transición de estados del NPC

Como podemos observar la única manera de llegar a los estados “Golpear” y “Huir”, esto es debido a que el NPC se volvía loco cuando no parabas de dispararle. Así que entraran en estos estados y cuando acaben con su cometido volverán a “Pasear”.

Para una mayor claridad pasamos a describir las clases de manera detallada:

Pasear

- Es el estado inicial, es decir, cuando se posiciona al NPC en el juego está por defecto en este estado.
- El NPC trazara una ruta por la universidad y caminara por ella.
- **Transición a “Golpear”**
  - Cuando el jugador choca o dispara al NPC y la variable aleatoria sale cero.
- **Transición a “Huir”**
  - Cuando el jugador choca o dispara al NPC y la variable aleatoria sale uno.

Ilustración 33: Estado NPC "Pasear"

Golpear

- El NPC se acercara a 100 unidades del jugador y lo golpeara.
- **Transición a “Pasear”**
  - Cuando haya perseguido al jugador durante 3 segundos o lo haya alcanzado y golpeado.

Ilustración 34: Estado NPC "Golpear"

Huir

- El NPC corre en dirección contraria al jugador.
- **Transición a “Pasear”**
    - Cuando haya huido del jugador durante 3 segundos.

Ilustración 35: Estado NPC "Huir"

Con esto el desarrollo del comportamiento de los NPCs estaría desarrollado. Ahora el juego tiene mucha más vida e interacciones que el simple hecho de ir a “pintar” al enemigo. Además que al generarse de manera aleatoria el número de NPCs cada partida tiene un toque distinto.

Para finalizar realizamos las últimas pruebas para asegurarnos que todo funciona correctamente. Una vez asegurado su pleno rendimiento pasamos a su evaluación, el cual se explicara en detalle en el apartado siguiente.

4. Pruebas y evaluación

4.1. Introducción

En este apartado se explicara la metodología que se ha seguido para evaluar el resultado final del desarrollo y comprobar si cumple con los objetivos fijados.

4.2. Explicación del método de evaluación

Analizar si se ha desarrollado correctamente una inteligencia artificial no es posible de manera técnica ya que es un comportamiento y lo que le transmite a los distintos jugadores. Así que para ello vamos a realizar un cuestionario con escala Likert [21], la cual nos permitirá saber cuánta satisfacción en una escala del 1 al 5 le ha producido al jugador.

La escala de Likert consiste en un cuestionario con preguntas que se contestan señalando un valor en una escala del 1 al 5, también se puede utilizar del 1 al 7 o del 1 al 10, las cuales hay afirmativas y negativas para poder analizar si el encuestado esta contestado con sinceridad o de manera aleatoria. Para obtener resultados coherentes tiene que dirigirse a un grupo de personas específico, en nuestro caso personas de entre 18 y 28 que ha jugado videojuego a lo largo de su vida de una manera más o menos constante.

El cuestionario que vamos a pasar a los 15 jugadores después de que prueben el juego es el siguiente:

		Muy desacuerdo	Desacuerdo	Ni desacuerdo ni de acuerdo	De acuerdo	Muy de acuerdo
ID	**Pregunta**	**1**	**2**	**3**	**4**	**5**
P1	A simple vista, nada más empezar el juego parecía divertido					
P2	El inicio del juego me ha parecido divertido					
P3	La interacción con los NPCs es divertida					
P4	La interacción con los NPCs es realista					
P5	Nada más abrir el juego me parecía aburrido					
P6	La interacción con los enemigos es divertida					
P7	La interacción con los enemigos es realista					
P8	En general el juego me ha parecido divertido					
P9	Tratar con los NPCs ha sido aburrido					
P10	Tratar con los enemigos ha sido aburrido					

Tabla 30: Cuestionario Likert

Como podemos observar las preguntas P5, P9 y P10 están en negativo, y por lo general preguntamos por la diversión y realismo del juego.

Para completar el cuestionario hemos realizado dos preguntas abiertas para saber en qué se puede mejorar y una opinión más general del encuestado. Las preguntas realizas como complemento son las siguientes:

Preguntas Complementarias	
¿Qué opinas en general de los NPCs y de los Enemigos?	
¿En qué ámbito podrían mejorar?	

Tabla 31: Preguntas Complementarias

Si los cuestionarios salen de manera correcta podremos hacernos una idea de cómo la inteligencia artificial ha beneficiado al juego en general.

4.3. Análisis de resultados

A continuación podemos observar las respuestas al cuestionario Likert de cada uno de los jugadores, se ha dividido en partes para una mayor claridad:

Respuestas al cuestionario. Parte 1				
P1. A simple vista, nada más empezar el juego parecía divertido	**P2. El inicio del juego me ha parecido divertido**	**P3. La interacción con los NPCs es divertida**	**P4. La interacción con los NPCs es realista**	**P5. Nada más abrir el juego me parecía aburrido**
4	3	5	3	2
5	4	4	3	2
3	3	4	2	3
3	2	3	3	3
4	2	4	3	2
4	2	4	4	2
5	3	5	4	1
3	2	4	3	3
3	1	3	2	3
3	2	3	2	2
4	3	4	4	2
3	3	4	3	3
4	2	4	3	2
4	3	5	3	2
3	3	4	3	2

Tabla 32: Respuestas al cuestionario. Parte 1

Respuestas al cuestionario. Parte 2				
P6. La interacción con los enemigos es divertida	**P7. La interacción con los enemigos es realista**	**P8. En general el juego me ha parecido divertido**	**P9. Tratar con los NPCs ha sido aburrido**	**P10. Tratar con los enemigos ha sido aburrido**
5	4	4	1	1
5	3	3	2	1
4	3	3	2	1
4	3	4	3	2
4	4	3	2	2
3	3	3	2	3
5	4	4	2	1
4	3	4	2	2
3	3	3	3	3
3	3	4	3	3
4	3	4	2	2
4	3	2	1	2
5	2	3	2	1
5	2	2	1	2
4	4	4	2	2

Tabla 33: Respuestas al cuestionario. Parte 2

Para verlo de una manera más clara convertimos las respuestas negativas en positivas, para observar la coherencia a simple vista, calculamos los porcentajes de cada escala, para saber la proporción que ha seleccionado en cada valor de la escala según la pregunta:

	P1	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	**P8**	**P9**	**P10**
Muy en desacuerdo	0,00%	6,67%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
En desacuerdo	0,00%	40,00%	0,00%	20,00%	0,00%	0,00%	13,33%	13,33%	0,00%	0,00%
Neutral	46,67%	46,67%	20,00%	60,00%	33,33%	20,00%	60,00%	40,00%	20,00%	20,00%
De acuerdo	40,00%	6,67%	60,00%	20,00%	60,00%	46,67%	26,67%	46,67%	60,00%	46,67%
Muy de acuerdo	13,33%	0,00%	20,00%	0,00%	6,67%	33,33%	0,00%	0,00%	20,00%	33,33%

Tabla 34: Porcentaje de Respuestas

Generamos un gráfico, ya que es la manera más clara y fácil de analizar unos datos, ya que dividiendo por colores desde verde la mejor hasta color rojo lo peor, se puede ver en un golpe de vista como ha sido la mayor contestación en cada pregunta, si positiva o negativa.

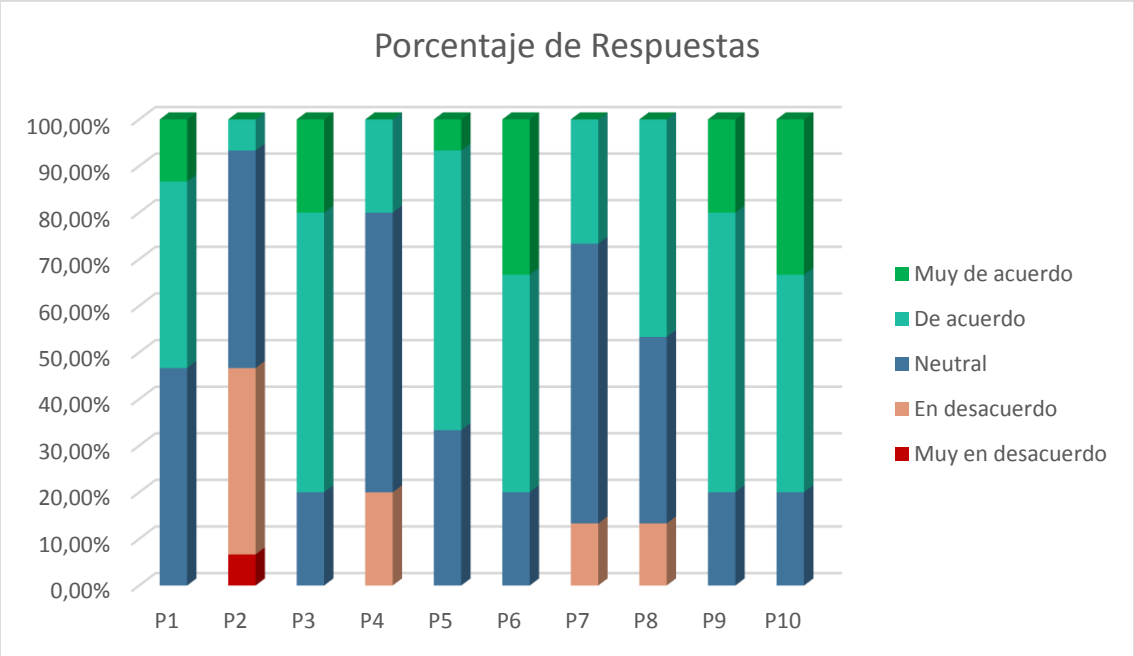


Ilustración 36: Porcentaje de Respuestas

Como podemos observar, de manera mayoritaria las respuestas han positivas aunque hay también un gran número de neutras. Por lo que podemos sacar que en general ha gustado pero que todavía se puede mejorar bastante, sobre todo al comienzo del juego ya que es la pregunta que más votos negativos se ha llevado.

Pero antes de seguir con el análisis vamos a comprobar con el alfa de Cronbach para verificar si este cuestionario es útil o no.

$$\alpha = \left[\frac{k}{k-1}\right]\left[1 - \frac{\sum_{i=1}^{k} S_i^2}{S_t^2}\right],$$

donde

- S_i^2 es la varianza del ítem i,
- S_t^2 es la varianza de los valores totales observados y
- k es el número de preguntas o ítems.

Ilustración 37: Alfa de Cronbach

Si el alfa da cerca de uno indicará que el cuestionario es fiable y cuanto más cerca de cero menos fiable.

Lo primero calculamos la varianza para cada pregunta, S_i^2:

S_1^2	S_2^2	S_3^2	S_4^2	S_5^2	S_6^2	S_7^2	S_8^2	S_9^2	S_{10}^2
0,488888889	0,51555556	0,4	0,4	0,32888889	0,51555556	0,38222222	0,48888889	0,4	0,51555556

Tabla 35: Varianza por pregunta

Ahora obtenemos la varianza de los valores totales, que consiste en sumar los valores de las respuestas de cada encuestado y hacer la varianza entre ellos:

S_t^2
17,9555556

Tabla 36: Varianza de los valores totales

Por lo tanto la ecuación quedaría así:

$$\left[\frac{10}{9}\right]\left[1 - \frac{4{,}4355556}{17{,}9555556}\right] = 0{,}836633663$$

El alfa nos sale mayor de 0,8 que es un valore que comúnmente es aceptado como correcto para dar como valido el cuestionario. Así que continuamos con el análisis, para ello generamos otro el porcentaje sobre el total de cada uno de los rangos:

Porcentaje sobre el total	
Muy en desacuerdo	0,67%
En desacuerdo	8,67%
Neutral	36,67%
De acuerdo	41,33%
Muy de acuerdo	12,67%

Tabla 37: Porcentaje sobre el total

Como indicamos anteriormente la respuesta más común ha sido positiva seguida de la neutral pero para ver con mayor claridad vamos a generar un gráfico para ver en correspondencia con las otras elecciones:

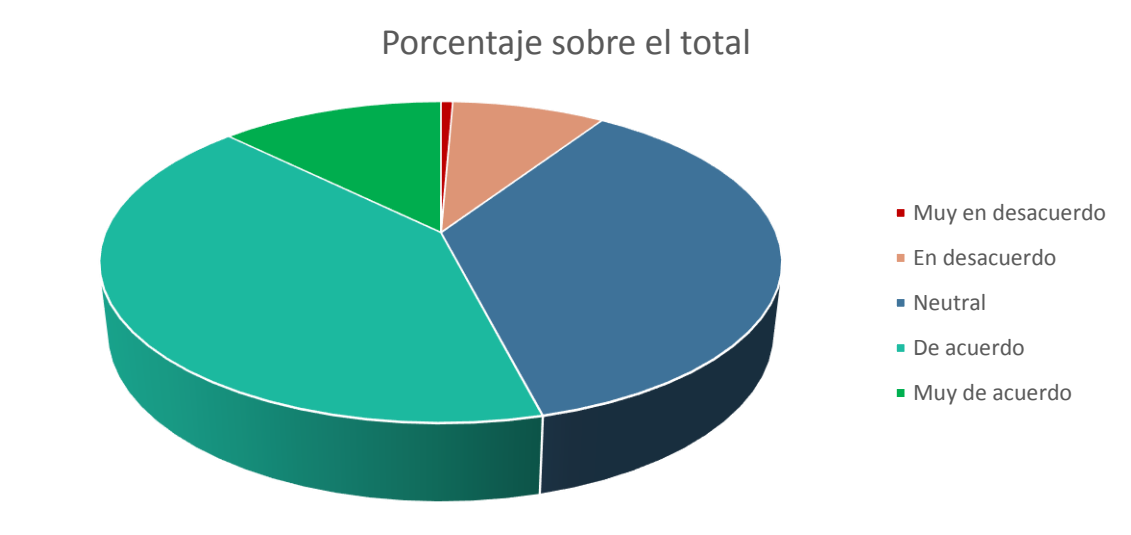


Ilustración 38: Porcentaje sobre el total

Se puede observar que más del 50% de las respuestas fueron positivas, teniendo en cuenta que en las preguntas negativas cuando se contestaba de manera negativa se ha cambiado para que se consideren positivas. Aun así casi el 40% son respuestas neutra, lo que indica que las cosas no están del todo mal pero pueden mejorar bastante.

Con un enfoque más individual podemos observar que en la pregunta P2 casi el 50% de respuestas han sido negativas. Esta pregunta hace referencia a la diversión nada más empezar el juego, en la cual el jugador tiene que ir a buscar el arma y a los enemigos. Esto indica que

esa parte es bastante aburrida y que tiene mucho ámbito donde mejorar, por ejemplo, que se un NPC el que le entregue el arma y le guie hasta los enemigos.

También para saber en qué ámbito se podía mejorar la inteligencia artificial de los enemigos y NPCs hemos creado las preguntas abiertas, de las cuales hemos sacado cinco ideas principales de cada una de las preguntas:

Conclusiones de las preguntas abiertas	
¿Qué opinas en general de los NPCs y de los Enemigos?	**¿En qué ámbito podrían mejorar?**
Son graciosos pero muy simples para lo que se suele ver en otros juegos	Podrían estar reunidos en grupos hablando para un mayor realismo
A veces los NPCs son molestos porque se cruzan en el tiroteo con los enemigos y vienen atacarte	Cuando se produzca un tiroteo entre los enemigos y el jugador los NPCs se aparten
Los NPC apenas tienen expresividad, no le cambia la cara para nada	Que la cara muestre más emociones
Los enemigos podrían cooperar más entre ellos para atacarte	Que los enemigos realizaran estrategias para ir a por ti
Los enemigos siempre salen en el mismo lugar por lo que es fácil encontrarlos	Que los enemigos salgan de diferentes puntos del mapa, que no estén todos juntos y que sean ellos desde el principio que vayan a por ti

Tabla 38: Conclusiones de las preguntas abiertas

Si analizamos las respuestas podemos observar que lo único referente al comportamiento de los NPCs y los enemigos es que son simples y no interaccionan entre ellos, ya que lo de las expresiones y el lugar donde aparecen los enemigos no es parte de la creación de la inteligencia artificial. Aunque para posteriores trabajos hay que tener muy en cuenta las recomendaciones aportadas por los encuestados ya que son bastante atractivas.

4.4. Mejora de la propuesta respecto a otras alternativas

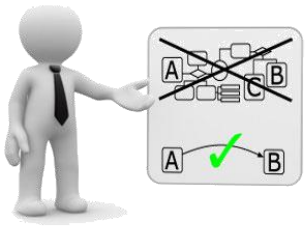


Ilustración 39: Mejora a otras alternativas

En este caso se ha demostrado que la máquina de estados ha sido la elección correcta ya que necesitábamos comportamientos básicos y muy acotados, y aunque la de búsqueda de caminos hubiera dado un resultado similar, el tiempo para realizarlo sería mayor por desconocimiento de cómo implementar esta idea y habría que empezar completamente de cero. Además que llevaría algún tiempo más de ejecución al ser una opción con mayor computacional. Y referente al aprendizaje por refuerzo, al igual que con la búsqueda de caminos, seria empezar de cero por no conocer cómo implementarlo en el UDK. Además del

tiempo que se tendría que utilizar para que la inteligencia artificial aprendiera el comportamiento adecuado, y habiendo acabado con un tiempo tan justo no hubiera sido factible esta opción.

Por lo tanto la máquina de estados ha sido más rápida en implementar, la que menor coste computacional tiene, por lo que la reacción se producirá en menor tiempo.

5. Aspectos económicos y legales

5.1. Introducción

En este apartado se va a detallar el coste del proyecto como si lo realizara el grupo encargado de la inteligencia artificial dentro de una empresa de videojuegos, por lo que solo tendremos en cuenta lo necesario para llevarlo a cabo.

5.2. Presupuesto

Para el cálculo del presupuesto se va a desglosar en costes personales, materiales y costes indirectos para un mejor análisis del mismo, además de un resumen para una mayor claridad y mayor comodidad para calcular el coste total.

En el presupuesto se realizara sobre las 14 semanas de trabajo las cuales se ha trabajado 30 horas por semana por lo que nos da un total de 420 horas de trabajo.

5.2.1. Personal

Para el coste de personal vamos a tener un jefe de proyecto, el cual organizara el equipo y repartirá las tareas, un analista, el cual se encargara del análisis y diseño sobre el papel, y un programador el cual llevara a cabo todo el proceso de creación. Para los salarios mensuales hemos realizado un pequeño estudio por internet y hemos obtenido que para un jefe de proyecto es 2000€, un analista 1600€ y un programador 1200€.

A continuación el desglose de gastos empresariales referentes al salario de cada uno de los trabajadores:

Cálculo de Costes Empresariales			
Datos del Empleado			
Puesto	Jefe de Proyecto	Analista	Programador
Jornada Laboral	1	2	3
Horas Semanales	5	10	15
Retribuciones Brutas			
Retribución Anual	3.000,00 €	4.800,00 €	5.400,00 €
Salario Base/Mes	250,00 €	400,00 €	450,00 €
Salario Neto/Año	3.500,00 €	5.600,00 €	6.300,00 €
Salario Cotizable	250,00 €	400,00 €	450,00 €
Pagas Extra	41,67 €	66,67 €	75,00 €
Base de Cotización	291,67 €	466,67 €	525,00 €
Impuesto sobre la Renta de las Personas Físicas			
Retención mensual IRPF	60,00 €	96,00 €	108,00 €
Retención anual IRPF	720,00 €	1.152,00 €	1.296,00 €
Retenciones de la Seguridad Social			
Contingencias Comunes (9,8%)	28,58 €	45,73 €	51,45 €
Accidentes Profesionales (1,8%)	5,25 €	8,40 €	9,45 €
Desempleo (6%)	17,50 €	28,00 €	31,50 €
Formación Profesional (0,6%)	1,75 €	2,80 €	3,15 €

Aportación FOGASA (0,4%)	1,17 €	1,87 €	2,10 €
Total Seguridad Social/Mes	54,25 €	86,80 €	97,65 €
Total Seguridad Social/Año	651,00 €	1.041,60 €	1.171,80 €
Salario Bruto Anual	**4.871,00 €**	**7.793,60 €**	**8.767,80 €**
Coste de Hora Trabajada	**20,30 €**	**16,24 €**	**12,18 €**

Tabla 39: Cálculo de Costes Empresariales

5.2.2. Material

Para los materiales tendremos en cuenta los equipos, que tienen que ser de gama alta para una ejecución de juego rápida y fluida, y el coste de amortización que necesitamos realizar. Además de algunos materiales fungibles para realización de tareas básicas.

Costes de Equipos				
Concepto	Modelo	Cantidad	Precio Ud.	Total
Ordenador Sobremesa	PcCom Gaming Battle A10-5800K	3	465,00 €	1.395,00 €
Monitor	LG 20M37A-B 20" LED	3	71,60 €	214,80 €
Total				**1.609,80 €**

Tabla 40: Costes de Equipos

Amortizaciones			
Concepto	Tiempo (Semanas)	Amortización Semanal	Base Imponible
Ordenador Sobremesa	14	99,64 €	1.395,00 €
Monitor	14	15,34 €	214,80 €
Totales		**114,99 €**	**1.609,80 €**

Tabla 41: Amortizaciones

Material Fungible			
Concepto	Cantidad	Precio Ud.	Total
Paquete Folios 500	1	2,50 €	2,50 €
Bolígrafos	6	1,00 €	6,00 €
Paquete de Post-It	3	0,80 €	2,40 €
Total			**10,90 €**

Tabla 42: Material fungible

5.2.3. Costes indirectos

Para los costes indirectos como son muy difíciles de calcular les daremos un valor del 5% de los costes totales ya que estos vienen derivados de los equipos y de las personas, ya sean desde la luz, el agua, el ADSL y otras necesidades puntuales.

El uso de un porcentaje fijo sobre los costes está bastante extendido y es una medida que ya se ha utilizado en proyectos anteriores con una muy buena aproximación.

5.2.4. Resumen de costes

Para una mayor claridad haremos un resumen de costes con los valores estrictamente necesarios para calcular nuestro precio final:

Gastos de Personal			
Datos del Empleado			
Puesto	Jefe de Proyecto	Analista	Programador
Horas Semanales	5	10	15
Total Semanas Invertidas	14	14	14
Coste Hora Trabajada	20,30 €	16,24 €	12,18 €
Total Coste	**1.420,71 €**	**2.273,13 €**	**2.557,28 €**

Tabla 43: Gastos de personal

Gastos Materiales	
Concepto	Coste
No fungible	1.609,80 €
Fungible	10,90 €
Total	**1.620,70 €**

Tabla 44: Gastos Materiales

5.2.5. Totales

Con estos costes y la realización de este trabajo en el tiempo de 420 horas obtenemos como precio final, incluyendo margen de seguridad de un 10% y margen de beneficios de un 15%, el valor de OCHO MIL TRESCIENTES TRES EUROS CON CINCO CENTIMOS.

Precio del Proyecto
Costes Empresariales
6.251,12 €
Costes Indirectos (5%)
6.563,67 €
Margen de Seguridad (10%)
7.220,04 €
Margen de Beneficio (15%)
8.303,05 €

Tabla 45: Precio del proyecto

5.3. Entorno socio-económico

Aunque para nuestro presupuesto apenas nos influye excepto en el salario de los trabajadores es importante tener en cuenta el estado socio-económico de España en este momento para la hora de ofrecer este proyecto y justificar su precio.

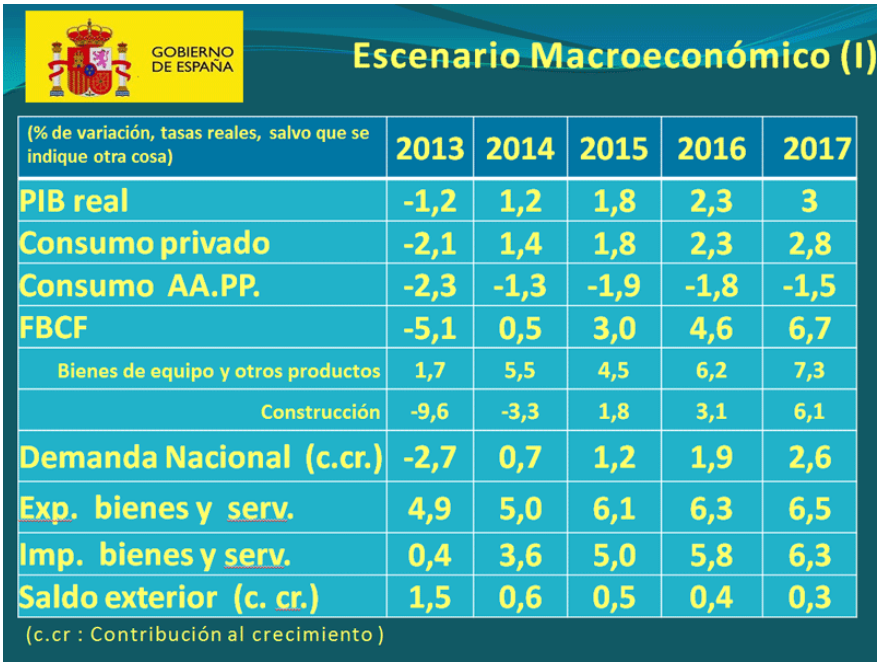


Tabla 46: Previsión del estado hasta 2017 Fuente: Gobierno de España

Este último año España a nivel económico está obteniendo mejores resultados en comparación a los 3 anteriores. Esto es debido principalmente de que otros países se están recuperando de la crisis y vienen de turismo e invierten más. También la falta de trabajo ha hecho que muchas personas se conviertan en autónomos y creen su propia empresa. Por lo que ha generado mucha competencia en sectores de alimentación o textil, ya que son trabajos sencillos que mucha gente puede realizar con una baja formación académica. Por lo contrario, en el sector informático se ha separado más que surgir nuevos negocios, y al ser un sector que requiere una formación específica sigue habiendo trabajo aunque no tanto como hace 3 años, y mucho menos igualmente pagado. Pero casi todas las empresas que se crean necesitan soporte informático ya sea desde páginas web hasta almacenamiento de datos. Y en el sector del entretenimiento cabe destacar los videojuegos ya que son las empresas que menos han estado perdiendo en la crisis y algunas han conseguido aumentar sus beneficios. Por lo tanto, es una situación favorable para presentar este proyecto ya que los videojuegos siguen creciendo y cada vez son más complejos.

5.4. Marco regulador

CLASIFICACIÓN DE PAÍSES CON MARCO LEGAL Y REGULATORIO MÁS FAVORABLE A LA COMPETITIVIDAD DE LAS EMPRESAS
-Ranking de 1 (más favorable) a 59 (menos favorable)-

Ranking	País	Ranking	País
1	Singapur	...	...
2	Hong Kong	29	Lituania
3	Malasia	...	...
4	Canadá	31	Austria
5	Suiza	...	...
6	Irlanda	34	Islandia
7	E. Árabes Unidos	...	...
8	Suecia	40	Turquía
9	Nueva Zelanda	41	Japón
10	Finlandia	42	Rumanía
11	Reino Unido	43	Rep. Checa
12	Australia	44	Portugal
13	Chile	...	...
14	Noruega	46	Bélgica
15	EE.UU.	47	Bulgaria
16	Dinamarca	...	...
17	Qatar	50	España
18	Alemania	51	Hungría
19	Estonia	52	Eslovenia
20	Países Bajos	53	Francia
...	...	54	Rep. Eslovaca
22	Luxemburgo	55	Grecia
...	...	...	...
25	Polonia	57	Croacia
26	Letonia	58	Italia

Fuente: Instituto de Estudios Económicos (IEE) a partir de "IWD Competitiveness Center", 2014

Tabla 47: Clasificación de países con marco regulatorio más favorable a la competitividad

Desgraciadamente a día de hoy no hay un marco legislativo para los videojuegos [22]. En la mayoría de los casos judiciales suelen tratarlo como software, pero no es totalmente correcto ya que un juego se compone también de música y arte. Pero a lo que hay que atenerse al desarrollar videojuegos, al menos hasta que cambie, es a trabajar al igual que si estuviéramos desarrollando software.

Ilustración 40: Copyright

También es importante a la hora de elegir qué tipo de licencia vamos a utilizar, ya que si utilizamos una comercial tendremos que tener en cuenta las leyes de propiedad intelectual y a que nos comprometemos para el usuario. Si en cambio cogemos una licencia libre, es

importante exactamente qué tipo es elegido ya que según la elección tendremos unos derechos y deberes distintos.

Ilustración 41: Creative Commons

6. Planificación del trabajo

6.1. Introducción

En este apartado se explicará la planificación que se tenía en un primer momento y luego la real, para finalizar analizando los motivos que han llevado a esas diferencias o porque han salido según lo deseado. Para ello, realizaremos unos cronogramas y un diagrama de Gantt en ambas planificaciones.

6.2. Planificación inicial

Para la planificación inicial se tiene pensado empezar el 2 de marzo hasta el 5 de junio, ya que la entrega seria el 9 de junio por lo que tendríamos cuatro días para cambios de última hora. Se trabajara en el proyecto un total de catorce semanas y se le va a dedicar seis horas diarias de lunes a viernes, es decir, treinta horas semanales. Por lo que se le va a dedicar un total de cuatrocientas veinte horas al proyecto.

6.2.1. Cronograma de actividades y control

Las tareas a realizar y como se repartirán se explica en el siguiente cronograma, en el cual se podrá ver la actividad las horas a dedicar, la fecha de comienzo y la fecha en la cual tiene que estar terminada.

Actividad	Tiempo estimado	Fecha de inicio	Fecha final
Análisis de objetivos	6 horas	02/03/2015	03/03/2015
Estudio del estado del arte	18 horas	03/03/2015	06/03/2015
Análisis de requisitos	24 horas	06/03/2015	12/03/2015
Estudio de técnicas para crear IA en videojuegos	18 horas	12/03/2015	17/03/2015
Desarrollo de las posibles alternativas	24 horas	17/03/2015	23/03/2015
Estudio de las ventajas e inconvenientes de cada alternativa	18 horas	23/03/2015	26/03/2015
Desarrollo del diseño elegido	210 horas	26/03/2015	14/05/2015
Pruebas	6 horas	14/05/2015	15/05/2015
Corrección de errores	18 horas	15/05/2015	20/05/2015
Evaluación	12 horas	20/05/2015	22/05/2015
Análisis de resultado de la evaluación	6 horas	22/05/2015	25/05/2015
Documentación	60 horas	25/05/2015	06/06/2015

Tabla 48: Cronograma de actividades inicial

6.2.2. Diagrama de Gantt

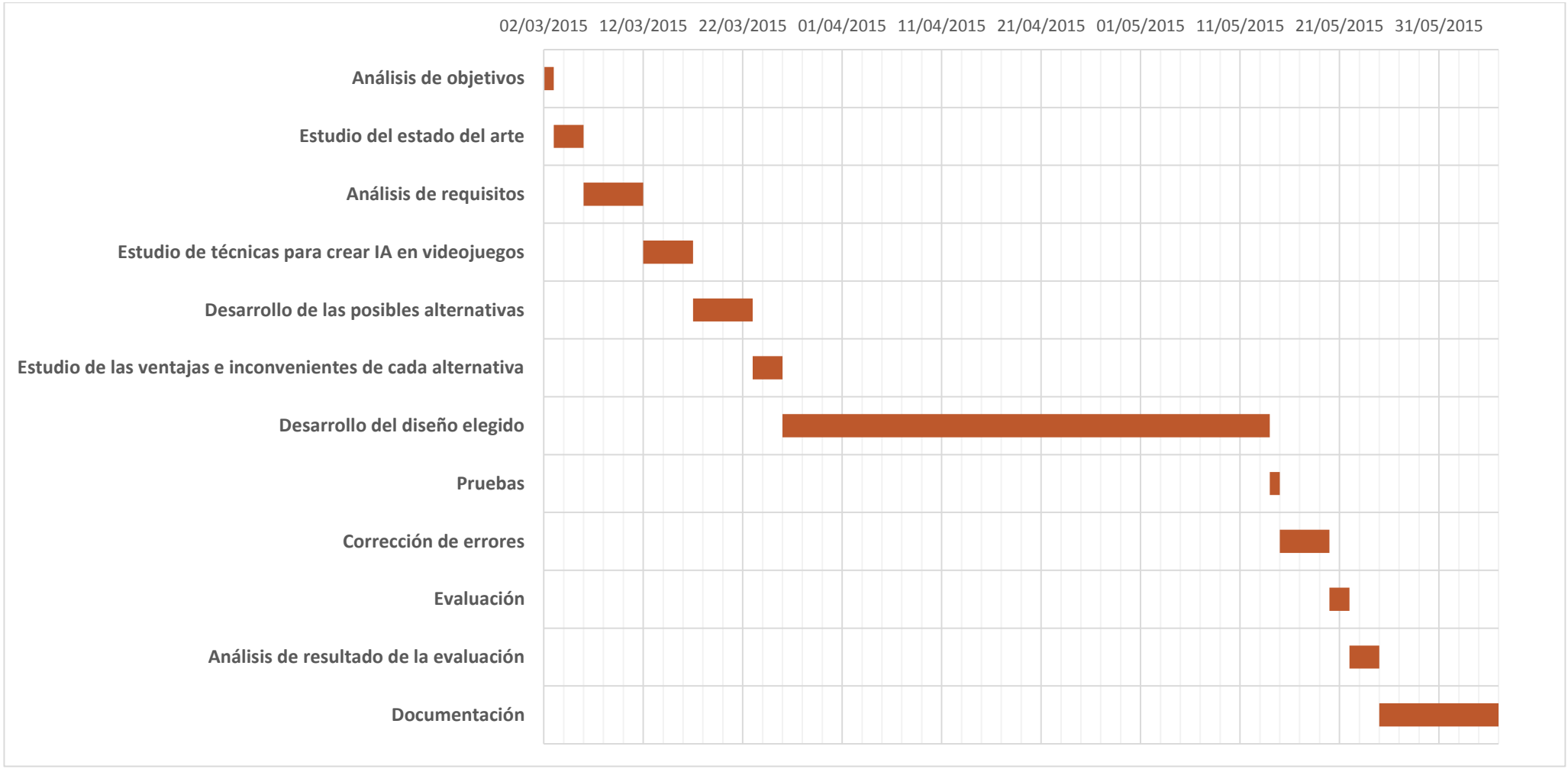


Ilustración 42: Diagrama de Gantt inicial

6.3. Planificación final

Debido a la realización de distintas prácticas, exámenes, y trabajo la entrega del trabajo quedó programada para la tercera sesión. Aunque se realizó algún avance en los meses anteriores se plantea la planificación desde que se empezó a realizar el proyecto de manera constante. El día que se empezó a realizar el proyecto de manera seria fue el 1 de junio. Se ha intentado dedicar de media 6 horas diarias pero al no poderlo asegurar se indicara en duración en días, también porque en el mismo día se han realizado distintas tareas. Al final el proyecto fue terminado el 11 de septiembre, por lo que podemos deducir que de manera aproximada se han realizado cuatrocientos cincuenta horas de trabajo, sin contar las que se realizaron con posteridad. Especificar que todos los días se fue trabajando en la memoria, por lo que le pondremos como tiempo de duración los setenta y cinco días que se ha trabajado en el proyecto.

6.3.1. Cronograma de actividades y control

Las tareas a realizar y como se repartirán se explica en el siguiente cronograma, en el cual se podrá ver la actividad las horas a dedicar, la fecha de comienzo y la fecha en la cual tiene que estar terminada.

Actividad	Tiempo duración	Fecha de inicio	Fecha final
Análisis de objetivos	3 días	01/06/2015	04/06/2015
Estudio del estado del arte	6 días	03/06/2015	11/06/2015
Análisis de requisitos	4 días	11/06/2015	17/06/2015
Estudio de técnicas para crear IA en videojuegos	6 días	15/06/2015	23/06/2015
Desarrollo de las posibles alternativas	6 días	18/06/2015	26/06/2015
Estudio de las ventajas e inconvenientes de cada alternativa	2 días	26/06/2015	30/06/2015
Desarrollo del diseño elegido	36 días	01/07/2015	20/08/2015
Pruebas	4 días	21/08/2015	27/08/2015
Corrección de errores	8 días	28/08/2015	09/09/2015
Evaluación	3 días	08/09/2015	11/09/2015
Análisis de resultado de la evaluación	1 día	10/09/2015	11/09/2015
Documentación	75 días	01/06/2015	11/09/2015

Tabla 49: Cronograma de actividades finales

6.3.2. Diagrama de Gantt

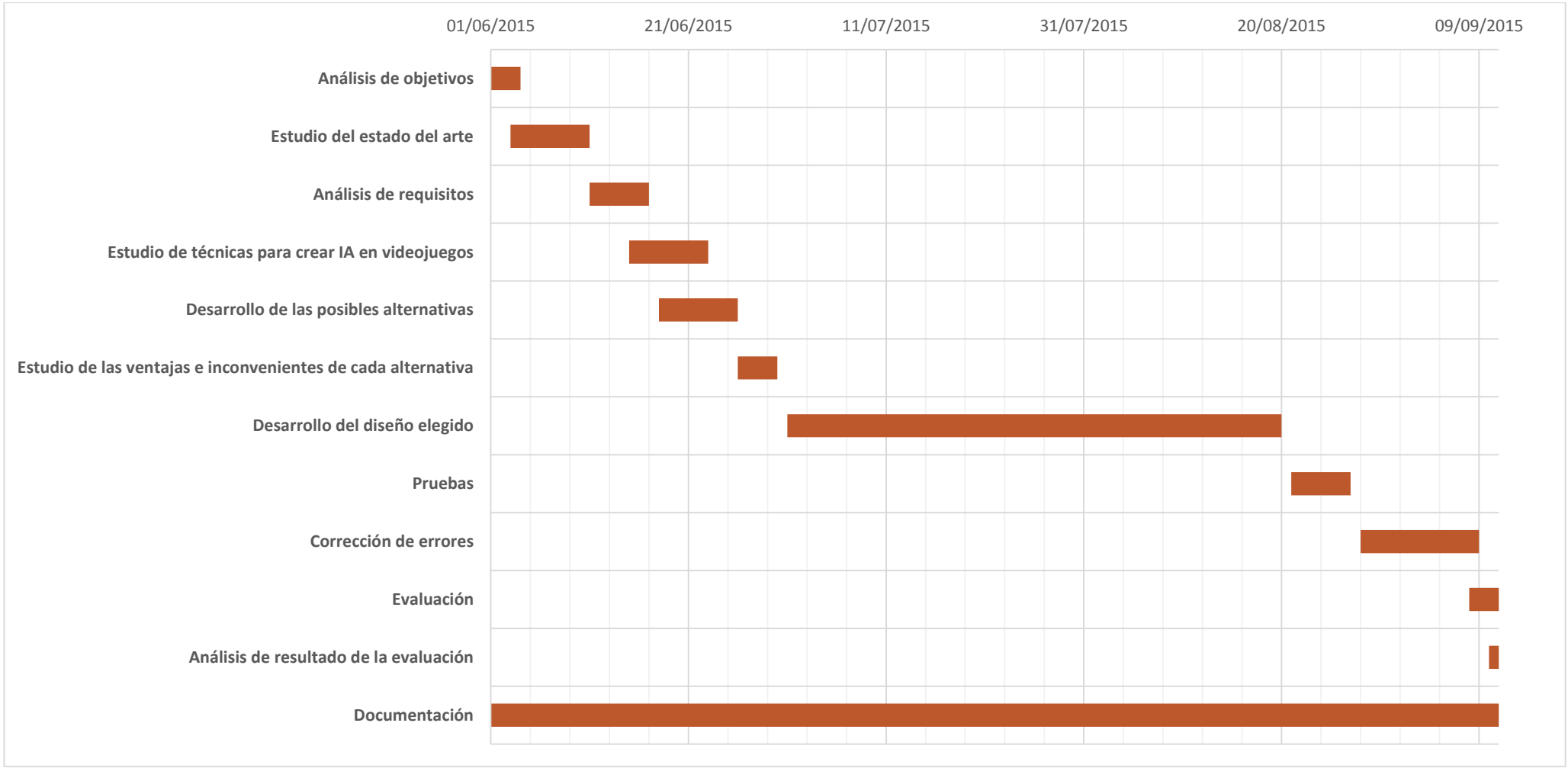


Ilustración 43: Diagrama de Gantt final

6.4. Comparativa del trabajo estimado y realizado

Como se puede observar a simple vista los planteamientos distan considerablemente, y si obviamos el tema de las fechas de inicio y fin podemos ver que también hay diferencias entre las tareas. Para una mejor visualización recurrimos a una gráfica que muestra la duración en días de ambos planteamientos, como la documentación se ha trabajado todos los días se ha realizado una estimación de cuantos días "enteros" se habría trabajado en ella para poder realizar una comparación adecuada:

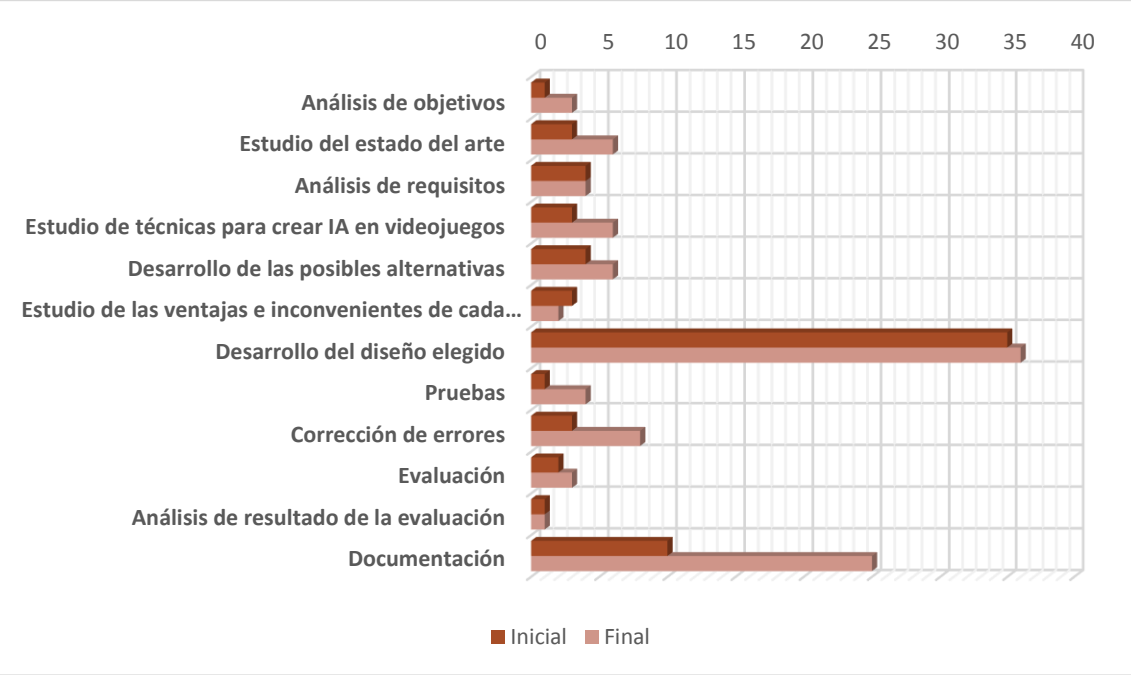


Ilustración 44: Diferencias en días del planteamiento inicial con el final

Si lo analizamos en detalle podemos observar que en el planteamiento final se han invertido más días, aunque también es debido a que todos los días también se trabajó en la memoria. Solo hay un caso en el que el final ha durado menos que el inicial y es en la actividad del estudio de ventajas e inconvenientes de las alternativas, esto resulto más fácil de lo esperado ya que a la hora de buscar las alternativas y documentarte, ya encontrabas las ventajas e inconvenientes de cada uno.

Por otro lado destaca la diferencia entre las actividades pruebas y corrección de errores, los cuales al final llevaron más tiempo del esperado porque al trabajar sobre el proyecto de otra persona a veces cuesta saber si el error viene del código nuevo o es el antiguo el que da problemas por las nuevas incorporaciones. Asique llevo mayor tiempo identificarlos y al haber más de los que se esperaban se tuvo que dedicar un mayor tiempo a solucionarlos. Cabe destacar que en la documentación también se ha invertido más tiempo del estimado, esto ha sido debido por que se ha trabajado directamente en ella y en vez de tomar notas y luego pasar a limpio.

7. Conclusiones

7.1. Introducción

En este apartado se analizara el resultado final para indicar cuanto de los objetivos se han completado, cuáles han sido los problemas, se estudiarán las líneas de futuro por las que podría seguir el proyecto y para finalizar se realizara una discusión sobre el proyecto, desde lo que ha aportado y lo que ha significado trabajar en él.

7.2. Objetivos cumplidos

El objetivo principal del proyecto era desarrollar una capa de inteligencia artificial básica para el comportamiento de los NPCs del videojuego *Paintball UC3M* (2013). Por lo que se puede considerar realizado ya que los enemigos y transeúntes de la universidad poseen una inteligencia artificial realizada con máquina de estados. Además los NPCs reaccionan según las acciones del jugador, por ejemplo, el enemigo si el jugador dispara, él se oculta; un transeúnte si es disparado por el jugado o huye o va a golpearlo. Por lo que podemos afirmar también que se ha conseguido una interactuación con el jugador y el responder a sus acciones.

A lo largo del proyecto han surgido objetivos imprevistos, tales como la búsqueda de un sistema de evaluación para nuestro proyecto, el cual se ha podido cumplir con satisfacción.

Lo único no conseguido, que se propuso en los requisitos como opcional por su gran complejidad, es que los NPCs interactúen entre ellos, algo que también se localizó en el apartado de evaluación y que los participantes también sugirieron.

7.3. Problemas encontrados

El mayor problema es el trabajar sobre el proyecto de otra persona ya que es un trabajo muy complejo y simplemente el proceso de entenderlo es muy largo y además no te aseguras que lo conozcas perfectamente. Para hacerse una idea se muestra una ilustración del Kismet del UDK, que es un esquema para realizar las acciones básicas del entorno y diferentes evento.

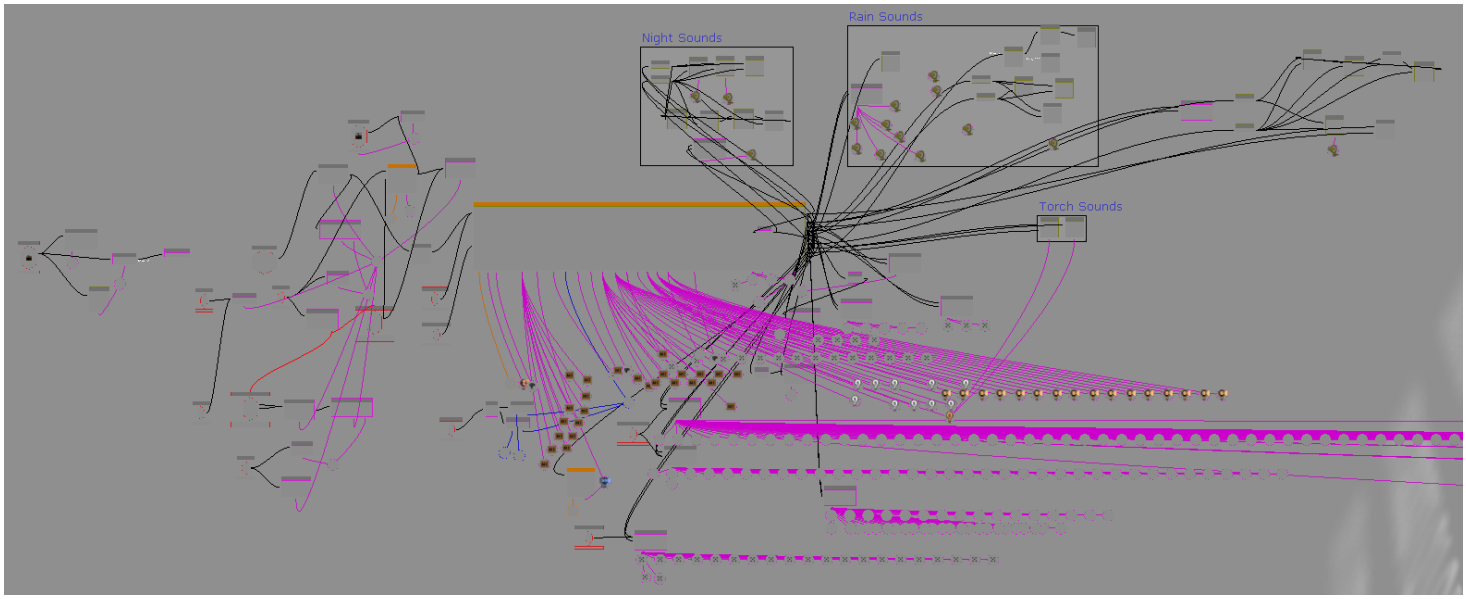


Ilustración 45: Kismet sin modificar del proyecto anterior

Como se puede observar es muy complejo y tiene una gran cantidad de datos, ya que controla lo del cambio del día a la noche, que haya lluvia o no, los diferentes sonidos, que al comienzo del juego el código lo sepa y empiece a funcionar, y un largo etc.

Además de la cantidad de código que se utiliza para su funcionamiento que se componía de dos partes, una de pruebas y otro que es el trabajo final. Se encontró que el trabajo final hacia también referencia a las clases de prueba algo que complico las cosas y se necesitó otro tiempo de análisis para configurar un esquema de funcionamiento. Después de todo este tiempo invertido solo en el estudio tienes el inconveniente de trabajar con el código de otra persona, que programa de forma distinta, y que además es escueta a la hora de comentar el código.

Otro problema fue el encontrar suficiente gente para la evaluación debido a que se hizo en el periodo de verano en el que mucha gente se fue de vacaciones y otros apenas disponían de tiempo.

7.4. Líneas futuras de trabajo

Debido por falta de tiempo o medios, la inteligencia artificial y el juego en general tiene mucho donde mejorar. Por lo que tenemos bastantes líneas de futuro para trabajar como:

- Como se ha ido analizando a lo largo del proyecto la posibilidad de que los NPCs, ya sean transeúntes o enemigos, interactuaran entre ellos. Por ejemplo que se reunieran en grupos para hablar o que jugaran al futbol, que el enemigo tenga en cuenta la posición de sus compañeros para atacar.
- Que los enemigos fueran aprendiendo las estrategias del jugador y las vaya utilizando contra él, de manera controlada para que el jugador no se viera asfixiado por la dificultad.
- Aliados NPCs, que acompañaran al jugador para enfrentarse a un enemigo más complejo o más “inteligente”.
- Después de un numero de oleadas podría salir un jefe, un enemigo que necesite más de un tiro y estrategia para ser vencido, con una inteligencia artificial complejo y que resulte a los jugadores todo un reto.
- Más armas y artefactos y que los enemigos sepan utilizarlos en cada momento e ir a buscar más.
- Generar más de dos bandos y contabilizar la victoria por puntos obtenidos, en los bandos los huecos que no sean ocupados por jugadores serán NPCs.

7.5. Conclusiones finales

Este proyecto a nivel personal ha sido todo un reto debido a que ya había utilizado UDK para proyectos personales, era la primera vez que trabajado con el proyecto de otra persona, además de que nunca había realizado una inteligencia artificial tan compleja, ni la de NPCs que tienen que deambular por un lugar y además interactuar con el jugador.

Y nunca hubiera llevado a cabo este proyecto si no hubiera sido por los conocimientos obtenidos en la carrera, ya que he tenido que recurrir constantemente a mis apuntes, sobre todo a los de Ingeniería del Software, Dirección de Proyectos de Desarrollo del Software y

Sistemas interactivos. Gracias a ellos he podido documentar correctamente todo el proyecto y me han guiado a la hora de su realización.

Por lo general, estoy orgulloso de mi trabajo ya que he podido demostrar todos mis conocimientos, las capacidades para obtener otros y adaptarme a todos los contratiempos que han ido surgiendo, tales como fecha límite, trabajos esporádicos y la controversia de interpretar el trabajo de otro. Además he demostrado que puede planificar un proyecto desde sus comienzos con el estudio de objetivos hasta su evaluación final.

Lo único que me arrepiento es de no haber tenido más tiempo para realizar muchas más cosas y haber realizado algunas de las líneas de futuro anteriores.

En definitiva, esta experiencia me ha otorgado los conocimientos necesarios para completar mis estudios como ingeniero informático y poder continuar mis estudios con el master el cual comienzo este 21 de septiembre para seguir aprendiendo.

Resumen en inglés

Introduction and objectives

Introduction

The AI or Artificial Intelligence is a very important point when we talk about videogames programming. It gives us the possibility of having different and completely independent player options as adversary, allied or just like an element to interact with.

Through the years this Artificial Intelligence is turning complex, at first it only described a simply performance like in the videogame called *Pong* (year 1972), in that game, the opponent actions only were defined by a little rules. Down to games like *Dragon Age: Inquisition* (2014) in which game the AI appears in the allied, enemies and other neutral characters that, keeping in mind the player actions, those characters act in a way or another.

In order to concentrate ourselves in the development of the Artificial Intelligence we will use the videogame created by Marco A. Pajares: *Paintball UC3M* (year 2013), in which game we are a player with a weapon that shoots paint who must finish the rivals, and the only AI present is in the enemies, that consist in following and shooting the player when they see him/her.

In this project we see the different use they give to the Artificial Intelligence in other videogames and which we'll choose for this specific case.

Personal motivation and objectives

My principal motivation comes since I was a child and I loved robotic, I used to build robots and see them moving, it was a challenge for me to see how with nothing at all we could create something that could move by itself. As the years went by I continued investigating because I couldn't be satisfied with the only walk of those robots. So I found some programming studies with I loved to know how with "nothing" we create "a lot". Then I discovered that one of my hobbies, the videogames, were closely related with programming. And not only that, there was a field about the performance, which reminds me of my beloved robots, and it was the Artificial Intelligence. With this project I'd like to go into detail about this field and take advantage of all the possibilities, it is something I wished a lot of time and now it's my opportunity to make it real.

The objective of this project is to develop a basic layer of Artificial Intelligence for the performance of the NPC (or non-player characters) of the videogame Paintball UC3M (year 2013). This layer of Artificial Intelligence is meant to be a concept proof which allows us to create an idea about the different possibilities the Artificial Intelligence can provide of dynamism and freshness to this videogame. Our principal objective is to develop some personalities for the NPCs (even if they are adversaries or just people which we can interact with). To make this possible, it will be necessary to analyze the current techniques, to create an appropiate design and modeling, to develop that layer to finally integrate it in the existent code and to prove its functionality.

Problem Statement

The problem we are going to deal with in this project is the incorporation of some NPC with a variety of Artificial Intelligences (from now on, we will call them “personalities”) in a current videogame.

The mission of those NPC is to give diversity and dynamism, as a person who walks through the environment who protests if we shoot him/her, or as an enemy who hide itself and search for the moment to shoot. Those elements are going to make the videogame more attractive and funny.

Memory Structure

This document is divided in seven parts which are specified and described below:

- **Problem statement**
  - We will treat from the art view of the Artificial Intelligence, the videogames and the connection between them. Also we treat the requirements the solution must resolve.
- **Design of technical solution**
  - We will analyze the different alternatives to solve the problem, valued from the advantages and disadvantages of each one. To finalize with this, there will be a detailed description of the development for the alternative chosen.
- **Tests and rating**
  - We will explain the evaluation method to use, we will analyze each one of the tests done and also we will study the possible improvements for all the problems discovered.
- **Economic and legal aspects**
  - We will refer to the estimated budgets for this project and the legal aspects to consider for the accomplishment.
- **Work planning**
  - We will develop an initial approach and a Gantt chart. Throughout the project we will show the real time of cost of each task with a comparison at the end, we will emphasize why our expectations are not fulfilled and how we must solve it the next time.
- **Conclusions**
  - To finalize the project we will analyze if the objectives are fulfilled, what changes we did and why, the future lines of work and a final conclusion about the project.

Development of the chosen design

To start the development of our solution, the first we must do is to get the six different models for our NPCs. To obtain that, we search some models online: http://www.cgtrader.com, http://tf3dm.com y http://www.turbosquid.com.

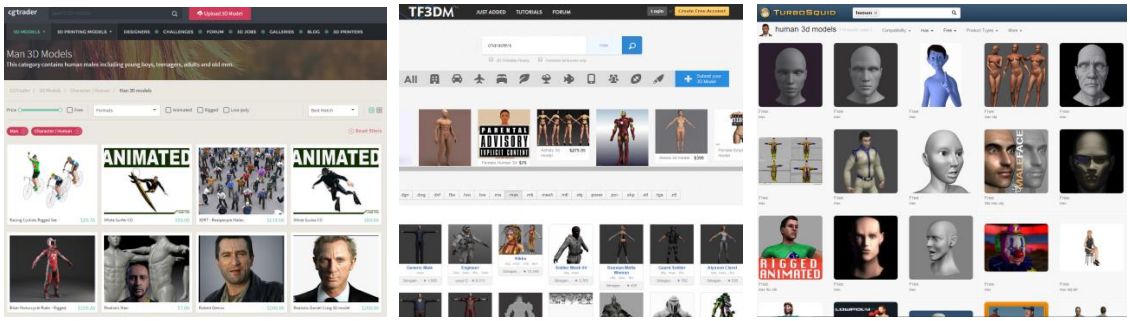


Ilustración 46: Páginas web de modelos 3D (Inglés)

There we find a huge variety of models but a little bit of that are really adapted for the game, even with that, we select some of them to make some tests and check if we can use them with the game engine. The first problem we find is that those models don't have a skeleton to animate them, this makes us loose time but we make a skeleton for one of that models with the 3D Max software because we have a previous experience with the use of this computer program:

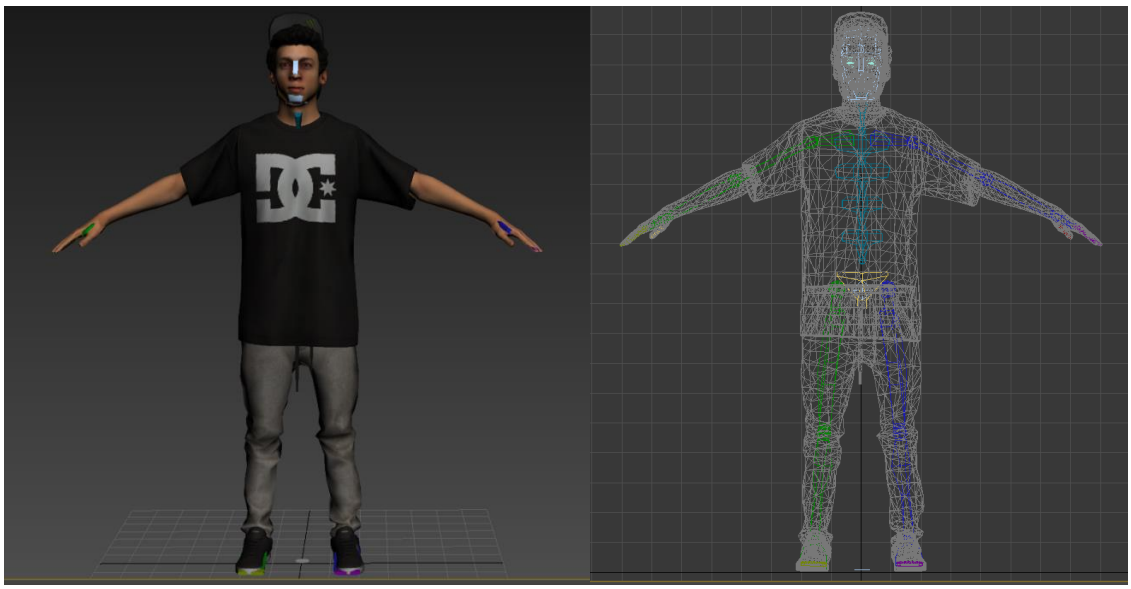


Ilustración 47: Modelo 3D con esqueleto (Inglés)

After this meticulous work, we export to the *UDK* motor to test the skeleton and in order to not have any problem with the placement of the bones which can deform our model when we animate it. In this test we check that the skeleton is not compatible with the default animations. We could try to put the default skeleton of the game engine to our models but it's complicated and less modifiable or flexible to the possible changes. But we investigate and we discover that the tool used in the project we are using for the base, hasn't disappeared (at a fist look is what we thought and because of that we didn't use like an alternative), that tool has been bought by Adobe and, in addition of creating 3D models, we can provide them a skeleton and there's a huge variety of compatible animations with them. So we decide to create our models with this application, *Mixamo*.

Ilustración 48: Modelo 3D creado con Mixamo (Inglés)

After we create the model we provide it a skeleton with the application and then we search some useful animations for our game.

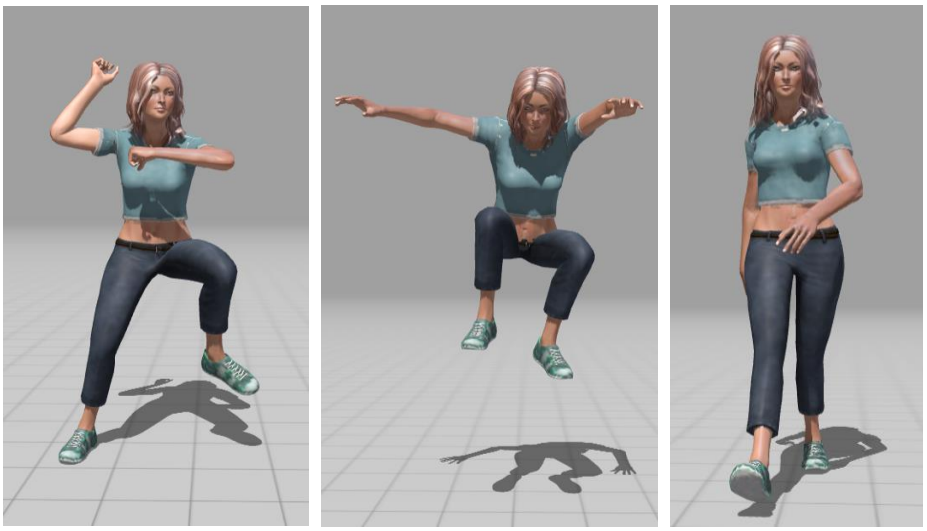


Ilustración 49: Animaciones (Inglés)

When we have it all the work, we export it to the *UDK* and we configure it to use them later from the code to design our state machine.

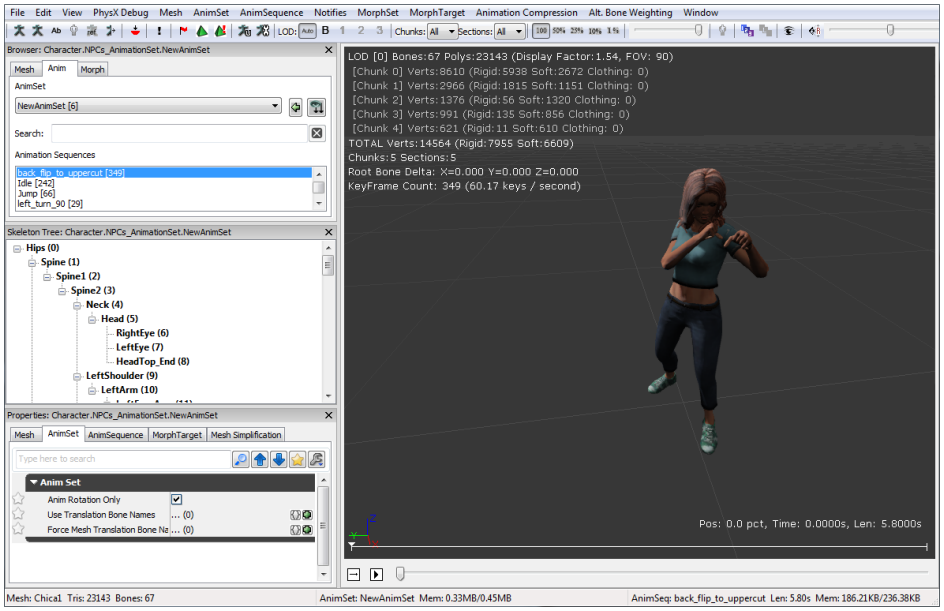


Ilustración 50: Configuración de animaciones en UDK (Inglés)

We repeat this process every time we want to create a model for the NPC. In our case, five more times to get a total of six different "persons" walking though the University.

To continue with our development we analyze the types the game is provided, for this we first create a relationship scheme and inheritance between them:

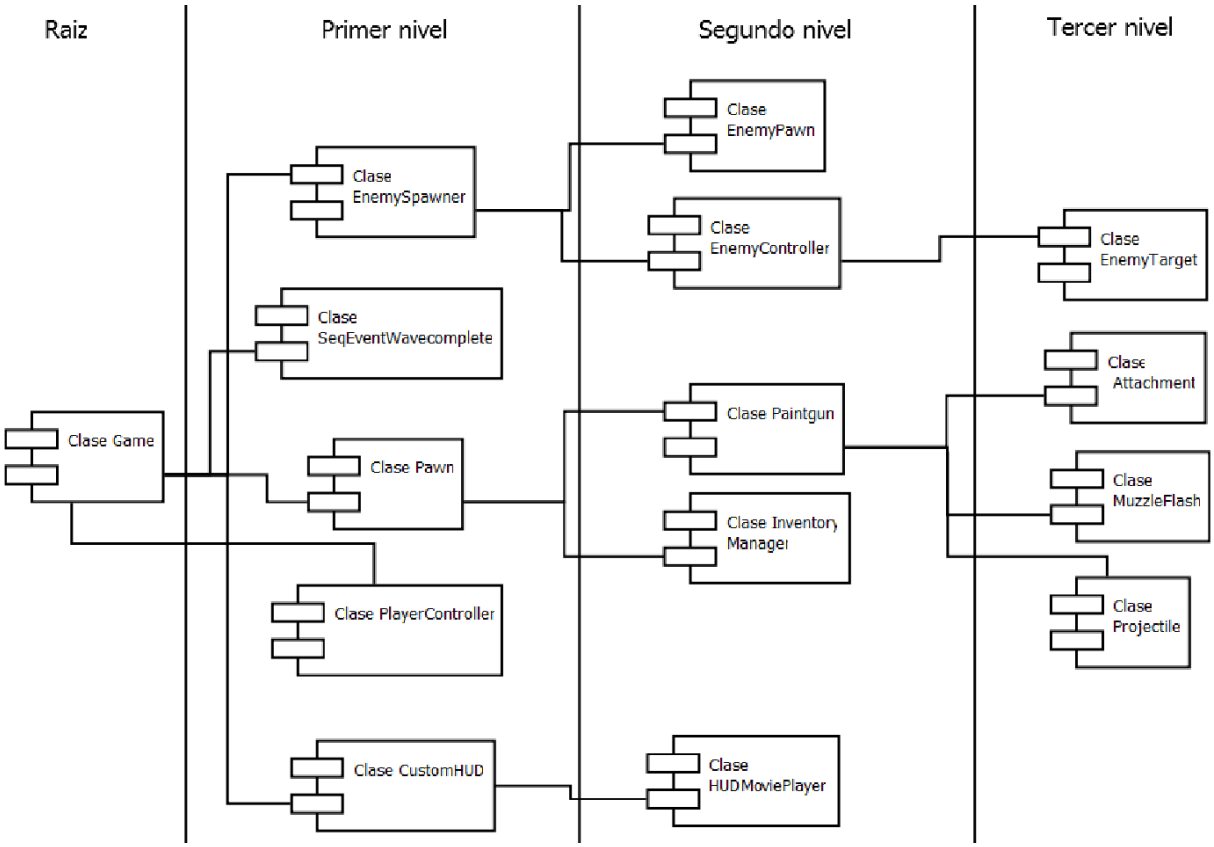


Ilustración 51: Relaciones y herencias entre clases (Inglés)

When we are clear about the level of inheritance and dependence, we start to analyze every type individually:

Clase Raíz	
Game	It implements the classes of persons we're going to use in the game instead of the default ones and it's the responsible to indicate if the game has finished or not.

Tabla 50: Clase Raíz (Ingles)

Clases del Primer Nivel	
EnemySpawner	It implements the creations of enemies in the game, leading to waves and the number of creations.
SeqEventWavecomplete	It implements the specification that indicates the game is finished.
Pawn	It implements the camera control, environment animations collisions and physical characteristics as the gravity or the speed of displacement.
PlayerController	It implements the player moves and its actions.
CustomHUD	It implements the new interface created for the game.

Tabla 51: Clases del Primer Nivel (Ingles)

Clases del Segundo Nivel	
EnemyPawn	It implements the first position of the enemies in the environment.
EnemyController	It implements the enemy behaviour.
Paintgun	It implements the drove of the paint weapon.
InventoryManager	It implements the management and storage of the objects.
HUDMoviePlayer	It implements the initial animation of the game.

Tabla 52: Clases del Segundo Nivel (Ingles)

Clases del Tercer Nivel	
EnemyTarget	It implements the action distance of the enemy.
Attachment	It implements the position where the character carries the weapon.
MuzzleFlash	It implements the system of particles of the paint weapon.
Projectile	It implements the trigger parameters.

Tabla 53: Clases del Tercer Nivel (Inglés)

After this analysis we can observe that we must to work in the type ***EnemyController*** to get the enemy behavior because is where the basic behavior is defined. To make this possible, in addition of the states "Mirando" in which they're searching for the player and "Persiguiendo" in which they're following the player, we're going to create the state "ACubierto" in which if the player shoots them and there's something to hide around, they'll use it. Another one is "Disparar" in which if the player is at a specific distance, the enemy will shoot. Therefor the state machines will be:

Enemy State Machine	
Mirando	State in which is searching for the player.
Persiguiendo	State in which the enemy run after the player.
ACubierto	State in which the enemy search a safe place.
Disparar	State in which the enemy shoots the player.

Tabla 54: Máquina de estados del enemigo (Inglés)

And the state transition, you're in a state which can change and at the same time we have no access.

State transition Enemy				
	Mirando	**Persiguiendo**	**ACubierto**	**Disparar**
Mirando	-	X	X	X
Persiguiendo	X	-	X	X
ACubierto	X	-	-	-
Disparar	X	X	-	-

Tabla 55: Transición de estados de Enemigo (Inglés)

So, when the model use the state "Mirando" it can change its state whenever the conditions were fullfilled. When the state is "Persiguiendo" and the model loses sight of the player it will turn to the state "Mirando" again, when the model gets fired in the state "ACubierto" and when it has within player's reach in the state "Disparar". When the model is in the state "ACubierto" and loses sight of the player it will be waiting since the player was again in the spotlight. In the state "Disparar", the model will stop to get more aim and this will provide the player the opportunity to walk away, in that moment it will change to the state "Persiguiendo" if the model continues watching the player or to the state "Mirando" if it can't.

A very important part is to calculate the time and the exactly conditions, for example, we found a problem that was about the player shooting to all the enemies and they hiding themselves, so if the player was constantly shooting the enemies didn't follow him/her. We had to retouch this and when the player shoot the enemies have a maximum time to get hide and they only hide if the player was far from them.

To clarify this we will define in detail every state:

Mirando

- The initial state, when the enemy is positioned in the game this is the default state.
- The enemy is waiting for the actions which make it change to another state.
- **Transition to “Disparar”**
  - When the player is in the offing and less of 400 unities of distance.
- **Transition to “ACubierto”**
  - When the player is in the offing and shoots, but only if the player wasn't more than 5 seconds in the state “ACubierto” and also more than 500 unities of distance.
- Transition to **“Persiguiendo”**
  - When the player is in the offing and UDK inform us with its variable.

Ilustración 52: Estado Enemigo "Mirando" (Inglés)

Persiguiendo

- The enemy is moving behind the player, following it.
- **Transition to “Mirando”**
    - When the enemy loses sight of the player.
- **Transition to “ACubierto”**
    - When the player is in the offing and shoots, but only if the player wasn't more than 5 seconds in the state “ACubierto” and also more than 500 unities of distance.
- **Transition to “Disparar”**
    - When the player is less of 400 unities of distance.

Ilustración 53: Estado Enemigo "Persiguiendo" (Inglés)

ACubierto

- The enemy searches the nearer place to hide and wait.
- **Transition to “Mirando”**
    - When the player is in the offing or is at least 5 seconds in that state.

Ilustración 54: Estado Enemigo "ACubierto" (Inglés)

Disparar

- The enemy stops and starts shooting the player with a 75% or more of success, considering distance.
- **Transition to “Persiguiendo”**
    - When the player moves away more than 400 unities of distance but still in the offing.
- **Transition to “Mirando”**
    - When the player moves away more than 400 unities of distance but is out of sight of the enemy.

With this, the enemy's behavior is developed and provides the game a better fun when we have to search and "paint" them.

Ilustración 55: Estado Enemigo "Disparar" (Inglés)

But we still have to develop the AI of the NPCs who will patrol the university to get naturalness and life to our environment. To make this possible, we have to start from scratch because the videogame didn't contemplate this possibility. The first step is to create the corresponding classes: ***NPCSpawner***, ***NPCPawn*** y ***NPCController***. All of them will be defined as in this table:

NPC's classes	
NPCSpawner	It implements the creation of the NPCs in the game, it takes charge of deciding which number generate.
NPCPawn	It implements the initial placement of the NPCs in the environment.
NPCController	It implements the NPCs behavior.

Tabla 56: Clases de los NPCs (Inglés)

To develop the classes we have to consider their namesake to the enemies. In ***NPCSPawn*** it doesn't generate waves and the number of NPCs will be decided randomly. To ***NPCPawn*** the NPCs don't generate at the same point but in different places defined by editor. And ***NPCController*** is the most different one because it defines a different behavior which will be controller for the next states machine:

NPC´s State Machine	
Pasear	State in which the NPC will travel the university.
Golpear	State in which the NPC walks to the player and hits it.
Huir	State in which the NPC walks away the player in the opposite direction.

Tabla 57: Máquina de estados del NPC (Inglés)

On this occasion the state "Pasear" will have different behaviors considering the situation, for example, if it rains the NPC will run to hide itself and if it grows dark it will go to the outskirts of the university, if these cases don't take place, the NPC will walk across the university like a regular person. The states "Golpear" and "Huir" will be activated as the same form but the decision to change to will be random.

Coming up next, we can see the transition table:

State transition of the NPC			
	Pasear	**Golpear**	**Huir**
Pasear	-	X	X
Golpear	X	-	-
Huir	X	-	-

Tabla 58: Transición de estados del NPC (Inglés)

As we can see, this is the only way to change to the states "Golpear" and "Huir" and this is because the NPC got crazy when you don't stop shooting it. So they change to these states and when they finish their mission they return to "Pasear".

To clarify this we will describe the classes in detail:

Pasear

- It's the initial state, when the NPC is positioned in the game this will be its default state.
- The NPC will set a course through the university and will walk by it.
- **Transition to “Golpear”**
    - When the player clashes or shoots the NPC and the random variable hits 0.
- **Transition to “Huir”**
    - When the player clashes or shoots the NPC and the random variable hits 1.

Ilustración 56: Estado NPC "Pasear" (Inglés)

Golpear

- The NPC will approach 100 unities to the player and will hit it.
- **Transición a “Pasear”**
    - When the NPC had followed the player during 3 seconds or it reached and kicked the player.

Ilustración 57: Estado NPC "Golpear" (Inglés)

Huir

- The NPC runs in the opposite direction of the player.
- **Transition to “Pasear”**
    - When the player has run away during 3 seconds.

Ilustración 58: Estado NPC "Huir" (Inglés)

With this, the development of the NPCs behavior is complete. Now the game is full of life and more interactions than the simple way of "paint" the enemy. Also we have another point because of the random generator of NPCs in every game.

To finish, we test to ensure that everything works correctly. When this is done we evaluate it and explain it below.

Conclusions

Introduction

In this paragraph we will analyze the final result to indicate how many objectives we have completed and which have been the problems, we will study the future ways our project could continue and to finish, the personal opinions of this project considering what I've contributed and what meaning has this for me.

Goals met

The principal objective of the project was to develop a layer of basic Artificial Intelligence to the behavior of the NPCs in the game *Paintball UC3M* (2013). We can consider it done because the enemies and passerby of the college has an Artificial Intelligence made with a state machine. Also the enemies react considering the actions of the player, for example, if the player shoots the enemy runs to hide; if a passerby is fired by the player, it can run away or go to kick him. So we can confirm that they have got an interacting with the player and the response to its actions.

The only thing they didn't have is that the NPCs don't interact with each other (this was an optional purpose on the requirements because of the complexity), this also was in the evaluation and the participants suggested.

Problems found

The biggest problem is working on the project of another person because this is a very complex work and it takes long time to just understand it and you can't know about it fully. To get an idea this is an illustration of the Kismet of the UDK, it's a schema to do the basic actions of the environment and different events.

As you can see, is very complex and it has a huge quantity of data, this is because it controls the day-night changes, the rain, the sounds… etc.

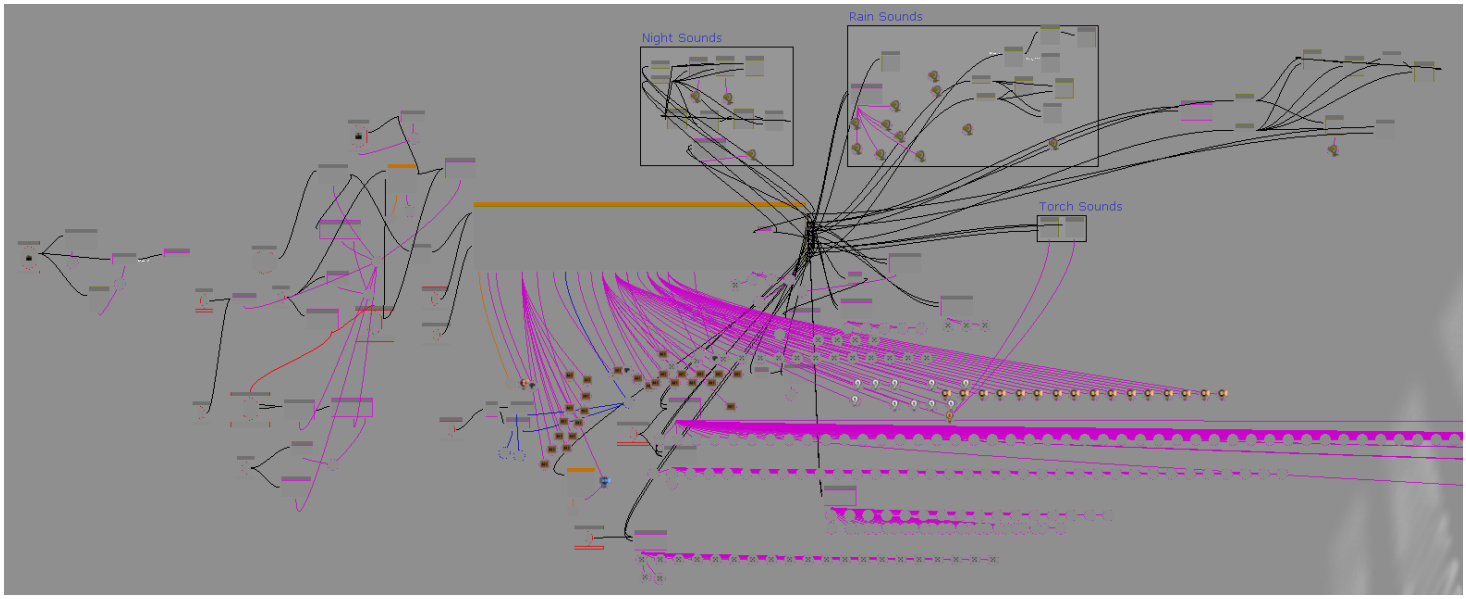


Ilustración 59: Kismet sin modificar del proyecto anterior (Inglés)

Besides the amount of code used for the correct performance and which consist of two parts, one part of tests and other which is the final work, we found that the final work was about the kind of tests, this was something complex and we needed some time to analyze and configure an operation scheme.

After this time spent in the study you have the inconvenient of working with the code of another person, that works on a different way and also who is short commenting the code.

Future lines of work

Due to lack of time or means, the Artificial Intelligence and the game in general have a lot to improve, so we have a lot of future lines to work like:

- How we were analyzing through the project the possibility the NPCs interact with each other's if they are passersby o enemies, for example, they had to join in group to talk or to play football or the enemy had to take in count the position of their partners to attack…
- The enemies had to learn the player's strategies and used them against him, but controllably in order to the player doesn't feel suffocated by pressure.
- The allied NPCs will follow the player to confront a complex enemy or more "intelligent".
- After a determinate number of waves a boss could appear, an enemy who need more than a shoot and a strategy to be defeated, with a complex Artificial Intelligence who must be a challenge for the players.
- More weapons and artifacts and that the enemies know in every moment how to use them and search more.
- To generate more than two proclamation and to count the victory considering the punctuation, the empty holes of the proclamation will be taken for NPCs if there's no player in that position.

Final conclusions

This project was a personal challenge because I used UDK for personal projects but this was the first time I worked based in the project of another person, also I've never done an Artificial Intelligence as complex as this one, nor the NPCs who must walk through a place and also interact with the player.

I never couldn't do this without the knowledge obtained in college because I had to appeal constantly to my notes, specially to the notes of Ingeniería del Software, Dirección de Proyectos de Desarrollo del Software and Sistemas interactivos. Thanks to them I could document correctly all the project and they guide me in the realization.

I'm proud of my work because I could demonstrate all my knowledge in it, the capacities to obtain others and adapt myself to all the encountered setbacks, so are the limit date, sporadic works and the controversy of interpret the work of other person. Also I demonstrated that one could plan a project from the beginning with the study of the objectives since the final evaluation.

Definitely, this experience has provide me the necessary knowledge to complete my studies as Computer Engineer and also the ability to continue my studies with a master which I start this future 21st of September to continue learning.

Bibliografía

[1] David WINTER. (2013). PONG-Story. Consultada el 2 de marzo de 2015, en http://www.pong-story.com

[2] BioWare. (2014). Dragon Age: Inquisition. Consultada el 2 de marzo de 2015, en http://www.dragonage.com/es_ES/home

[3] M. A. PAJARES, "Desarrollo de un videojuego con udk," Trabajo de fin de grado, Ingeniería Informatica, Universidad Carlos III de Madrid, Leganés, 2013.

[4] Julio Vilena Román, Raquel M. Crespo Garcia y José Jesús García Rueda. (2012). Historia de la Inteligencia Artificial. Consultado el 5 de marzo de 2015, en http://ocw.uc3m.es/ingenieria-telematica/inteligencia-en-redes-de-comunicaciones/material-de-clase-1/01-historia-de-la-inteligencia-artificial

[5] ITAM. (2015). Breve historia de la Inteligencia Artificial. Consultada el 4 de mayo de 2015, en http://biblioteca.itam.mx/estudios/estudio/estudio10/sec_16.html

[6] Jose M.ª Viani Sallaberry. (1995). Sinopsis histórica de la inteligencia artificial. Consultada el 4 de mayo de 2015, en http://www.actuarios.org/espa/revista12/17-art08.pdf

[7] Grupo 5 - RAI - UC3M. (2012). Redes de Neuronas Artificiales. Consultada el 6 de mayo de 2015, en http://www.lab.inf.uc3m.es/~a0080630/redes-de-neuronas/index.html

[8] José Manuel Abad Liñán. (2015). El robot niño aprende solo. Consultada el 1 de junio de 2015, en http://tecnologia.elpais.com/tecnologia/2015/06/03/actualidad/1433328081_377500.html

[9] Nadala Fernández y Pedro Hugo Carmo. (2008). Historia de los videojuegos. Consultada el 2 de junio de 2015, en http://www.fib.upc.edu/retro-informatica/historia/videojocs.html

[10] Javier Sotillo Mallo y Cristian Martínez Ruiz. (2014). Inteligencia Artificial en los Videojuegos. Consultada el 8 de junio de 2015, en http://www.it.uc3m.es/jvillena/irc/practicas/13-14/03.pdf

[11] Alex J. Champandard. (2007). Top 10 Most Influential AI Games. Consultada el 9 de junio de 2015, en http://aigamedev.com/open/highlights/top-ai-games/

[12] Alex J. Champandard. (2014). Games of the Year: The 2014 AiGameDev.com Awards for Game AI. Consultada el 9 de junio de 2015, en http://aigamedev.com/open/editorial/2014-awards/

[13] José Carlos Cortizo Pérez. (2010). IA en Videojuegos. Consultada el 10 de junio de 2015, en http://es.slideshare.net/jccortizo/ia-videojuegosd

[14] Tim Schreiner. (2013). Artificial Intelligence in Game Design. Consultada el 10 de junio de 2015, en http://ai-depot.com/GameAI/Design.html

[15] Oliver Barraza. (2011). Importing Mixamo animations into Unreal Development Kit. Consultada el 22 de junio de 2015, en https://www.mixamo.com/workflows/udk

[16] Alex Galuzin. (2012). UDK: How to Spawn Bots in Kismet. Consultada el 6 de julio de 2015, en http://www.worldofleveldesign.com/categories/wold-members-tutorials/petebottomley/udk-01-how-to-spawn-bots-in-kismet.php

[17] Marcos Díez. (2013). PFM: Creación de un generador de enemigos en UDK. Consultada el 20 de julio de 2015, en https://marcosdiez.wordpress.com/2013/01/14/pfm-creacion-de-un-generador-de-enemigos-en-udk/

[18] Marcos Díez. (2013). PFM: Scripting de las oleadas de enemigos con Kismet. Consultada el 20 de julio de 2015, en https://marcosdiez.wordpress.com/2013/02/13/pfm-scripting-de-las-oleadas-de-enemigos-con-kismet/

[19] Marcos Díez. (2013). PFM: Diseño y creación de enemigo Speeder. Consultado el 3 de agosto de 2015, en https://marcosdiez.wordpress.com/2013/01/30/pfm-diseno-y-creacion-de-enemigo-speeder/

[20] Cédric Hauteville. (2012). UDK – AI Pawn movement. Consultada el 5 de agosto de 2015, en http://www.moug-portfolio.info/udk-ai-pawn-movement/

[21] Néstor Malave. (2007). Trabajo modelo para enfoques de investigación acción participativa programas nacionales de formación. Consultado el 17 de agosto de 2015, en http://uptparia.edu.ve/documentos/F%C3%ADsico%20de%20Escala%20Likert.pdf

[22] Ruth Castellote. (2013). ¿Es un problema la clasificación legal de los Videojuegos?. Consultado el 24 de agosto de 2015, en https://ruthcastellote.wordpress.com/2013/12/04/es-un-problema-la-clasificacion-legal-de-los-videojuegos/

Acrónimos, abreviaturas y definiciones

UDK: Unreal development Kit, entorno de desarrollo de videojuegos.

IA: Inteligencia Artificial.

AI: Artificial Intelligence, abreviatura en ingles de Inteligencia Artificial.

NPC: non-player character, abreviatura en ingle de personaje no manejado por el jugador.

NPCs: Término popular e incorrecto del plural de NPC (observar definición arriba), el término correcto y que no se suele utilizar seria NPCC.

FNC: Requisito funcional.

NFN: Requisito no funcional.

UC3M: Universidad Carlos III Madrid.

HUD: Es el interfaz que en el juego le da la información al jugador, tales como la vida y la munición.

- Fin del documento -